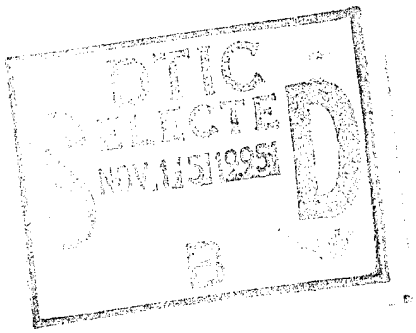
Naval Research Laboratory
Monterey, CA 93943-5502

NRL/MR/7543--95-7220

User's Guide and Documentation for the NRL Environmental Data Browsing System (NEBS)

GARY G. LOVE

Forecast Support Branch
Marine Meteorology Division



September 1995

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. Agency Use Only *(Leave blank).*	2. Report Date.	3. Report Type and Dates Covered.
	September 1995	Final

4. Title and Subtitle.
User's Guide and Documentation for the NRL Environmental Data Browsing System (NEBS)

5. Funding Numbers.
PE 0603207N
PN X0514
AN DN153165

6. Author(s).
Gary G. Love

7. Performing Organization Name(s) and Address(es).
Naval Research Laboratory
Marine Meteorology Division
Monterey, CA 93943-5502

8. Performing Organization Report Number.
NRL/MR/7543--95-7220

9. Sponsoring/Monitoring Agency Name(s) and Address(es).
Space and Naval Warfare Systems Command (PMW-175)
Washington, DC 20363-5100

10. Sponsoring/Monitoring Agency Report Number.

11. Supplementary Notes.

12a. Distribution/Availability Statement.
Approved for public release; distribution unlimited.

12b. Distribution Code.

13. Abstract *(Maximum 200 words).*

The NRL Environmental Data Browsing System (NEBS) is a graphical user's interface (GUI) designed to visualize 4D (x,y,z,t) gridded field data from numerical models to help evaluate and diagnose model behavior. It allows the user to quickly slice and animate data displays in x,y,z, or t and easily reveals anomalies and erroneous trends. Setting dimensions at any identified problem's conditions provides a quantitative listing of individual variables. Variables and dimensions can be easily set using mouse button clicks on the displayed GUI buttons. The NEBS uses the Grid Analysis and Display System (GrADS) developed by the Center for Ocean-Land-Atmosphere Interactions (COLA) at the University of Maryland. NEBS can be used to browse almost any data in the GrADS IEEE 4* real binary format and other data formats supported by GrADS (e.g., gridded binary format (GRIB)). NEBS and GrADS are supported on a variety of Unix platforms including Digital, IBM, Hewlett-Packard, NeXT, Silicon Graphics, Sun, and Concurrent. Graphics are displayed quickly and can be animated on a host platform at about one frame per second or about one frame per two seconds over a high speed LAN.

14. Subject Terms.			15. Number of Pages.
Data browser	2D graphics	Animation	84
GUI	GrADS		16. Price Code.

17. Security Classification of Report.	18. Security Classification of This Page.	19. Security Classification of Abstract.	20. Limitation of Abstract.
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	Same as report

Contents

User's Guide and Documentation for the NRL Environmental Data Browsing System (NEBS)

1. **Introduction.** The NRL Environmental data Browsing System (NEBS) was developed to quickly visualize 4D (x, y, z, t) gridded field data from numerical models and to help evaluate and diagnose model behavior. The NEBS allows the user to quickly slice and animate data displays in x, y, z, or t. NEBS was specifically developed to support the Variability of Coastal Atmospheric Refractivity (VOCAR) experiment, but was designed to be very general. NEBS can be used to browse almost any data formatted in accordance with the specifications in Appendix A. The script for the NEBS is listed in Appendix B.

2. **General Features.** The NEBS uses the Grid Analysis and Display System (GrADS; Doty and Kinter, 1992) developed by the Center for Ocean-Land-Atmosphere Interactions (COLA) at the University of Maryland. NEBS is written in the GrADS scripting language which GrADS interprets and executes. GrADS is supported on a variety of platforms including DOS-based PC and Unix workstations (Digital, IBM, Hewlett-Packard, NeXT, Silicon Graphics, Sun, and others). However, the NEBS is currently only supported on the Unix workstations.

2.1 Data Graphical User Interface. The NEBS is a graphical interface designed for quick-look and diagnostics of model output. It reads data in the GrADS data format and other data formats supported by GrADS, such as GRIB. Simple translation programs are available to reformat NORAPS and NOGAPS files.

2.2 Mouse and Button Control. Browsing is controlled entirely with a mouse cursor, mouse buttons, and GUI "buttons" to select options and execute commands.

2.3 Selectable Variables. A button for each variable in a data set is created automatically when a data file is selected. Multiple variables can be selected and displayed simultaneously as contours. One variable, at most, can be shaded.

2.4 Selectable Displays. Vector variables can be displayed as streamlines, vectors, or barbs. Contour intervals and limits can be customized for each variable, and vector variables can be parsed, if too dense.

2.5 Selectable Dimensions and Times. Spatial dimension and time bars appropriate for the data contents of the selected file are displayed and allow the selection of precise levels or varying ranges in each dimension or time (only three of these four parameters can vary simultaneously).

2.6 Selectable Stills or Animation. Maps, cross sections, or line plots are displayed, or animated, based on the selected dimensional conditions. This allows the data to be sliced and animated in X, Y, Z, or T. Animation occurs if the indicated dimension or time bars have highlighted ranges.

2.7 Time Interpolation. Time animations can be more highly resolved by selecting a time interval to interpolate the data set values.

2.8 Zoom and Pan. Zoom in, zoom out, and panning of map displays of model data can be used for a quick investigation of data anomalies.

2.9 Selectable Reference Observations. Radiosonde profiles and surface observations can be compared to model forecasts for evaluation.

Profiles. Raob data and model profiles can be displayed simultaneously for only a single station. The profiles can be time animated, but if time interpolated, the raob data only appears at its valid time.

Time Lines. Maps of selected model variables and observed surface winds can be displayed and animated. If time interpolated in a map view, both model variables and surface observations are interpolated and displayed simultaneously. Individual surface stations can be independently selected in the map view to allow the display of selected model variables and surface observations as time lines for each station.

2.10 Selectable Output Media. Once a single display or animation sequence is specified, the frame(s) can be sent to a black/white printer, color printer, GrADS movie file, or video editor (not currently implemented) for creation of a VHS video tape.

3. Installation and Setup.

If GrADS is not already installed, it will be necessary to download Version 1.5, or later, from COLA at the University of Maryland (grads.iges.com, IP# 192.239.84.50).

If operating remotely over a LAN, setup the local xhost to accept a remote display using the 'xhost +' command on your local workstation and designate your *workstation* to the xserver using:

setenv DISPLAY *workstation*:0.0

The *directory* on the remote xserver where GrADS is installed must be added to your search PATH and the environment must be set as follows in your login file:

setenv TERM xhost
setenv GADDIR *directory*

The GrADS data control files for the data to be browsed must be in the same directory as the NEBS script. If station raob and surface data files are used, they must be accompanied by one file each that specifies the station names and locations in each observed data file; named 'stations.ua' and 'stations.sfc', respectively. The formats for these files are described in Appendix A. This naming convention is necessary unless the NEBS script is modified. GrADS control files for model data may have any name, but must have the standard ".ctl" extension.

4. Using the NRL Environmental Browsing System.

4.1 Start Up. First, open an xterm window, change directories to the directory containing the browser, and initiate GrADS by typing

grads -l {Enter}.

At the GrADS prompt (i.e., ga>), type

run nebs.gs {Enter}.

Alternatively, these can be combined into a single command in the xterm window by typing

grads -lc "run nebs.gs" {Enter}.

A list of available data files will appear. A file from this list may be typed in response to the query for a grid data filename. (This may also be accomplished with 'copy' and 'paste' operations, if these features are available on your workstation.) Next, a query will appear requesting the surface data filename, followed by a query for the raob data filename. If surface or raob data are not available, or are not desired, an {Enter} will bypass these queries. The browser will start up, as shown in Figure 1, by displaying a window with control buttons for selecting variables, graph types, time settings, dimension settings, and basic functions, such as reset, print, quit, and display.

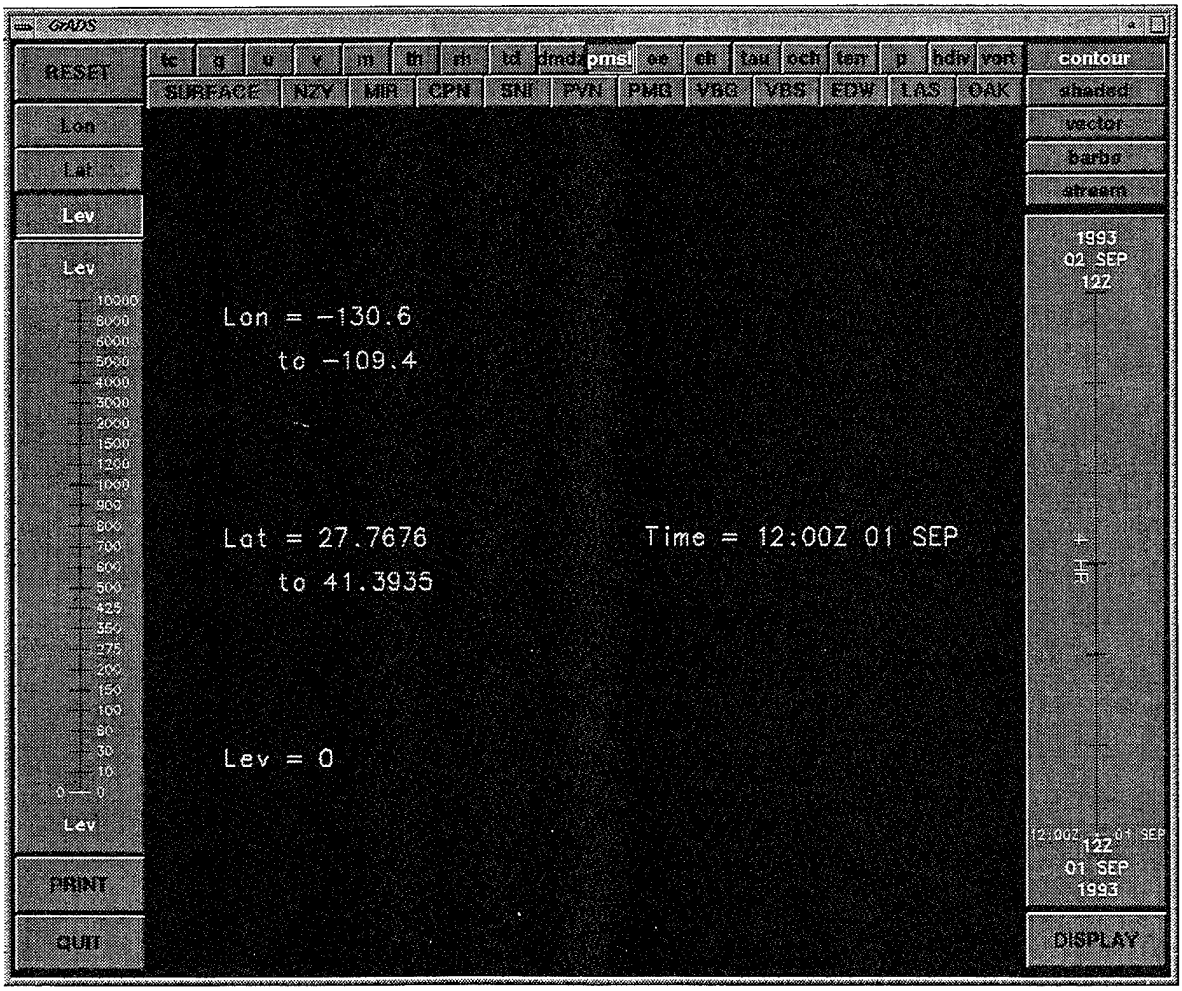


Figure 1. Basic NEBS control buttons.

4.2 Variable Selection. Variables can be "toggled" off and on by placing the mouse cursor on the desired variable's button and by clicking the left or middle mouse button. Also, the browser will start automatically with sea level pressure or surface pressure displayed (Fig. 2), if present in the data set, to provide an initial orientation to the meteorological conditions existing at the first time step.

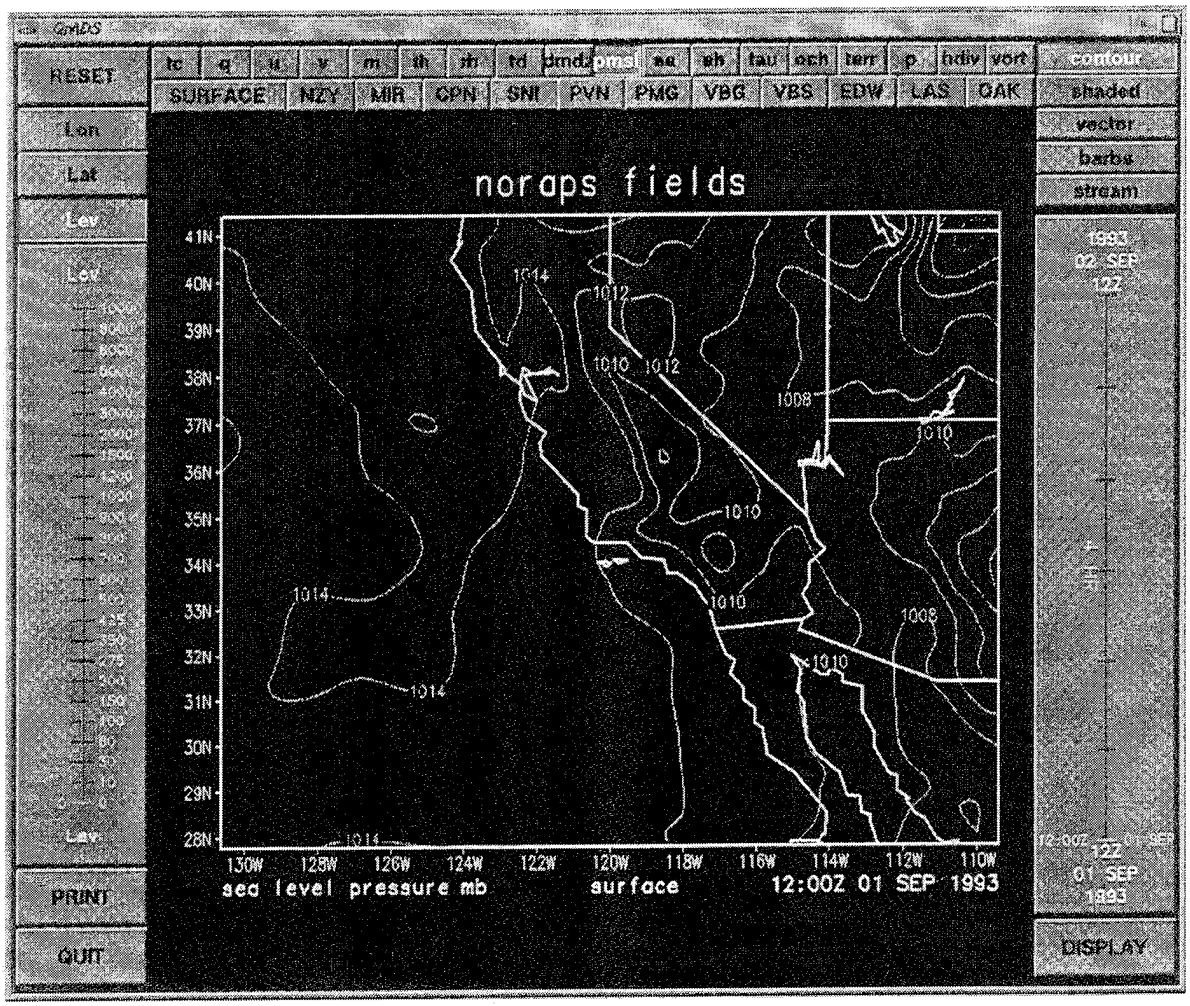


Figure 2. Initial pressure display (when data present).

4.3 Variable Customization. A menu of options (Fig. 3) can be obtained by placing the mouse cursor on the desired variable's button and by clicking the right mouse button. All options must be selected using the right mouse button. Clicking the left or middle mouse button accepts the displayed menu values.

4.3.1 Contour labels. The presence of contour labels can be toggled on and off by clicking the On/Off status button.

4.3.2 Contour interval. The plot contour (or shading) interval can be incremented or decremented by clicking the '>' or '<' symbol, respectively.

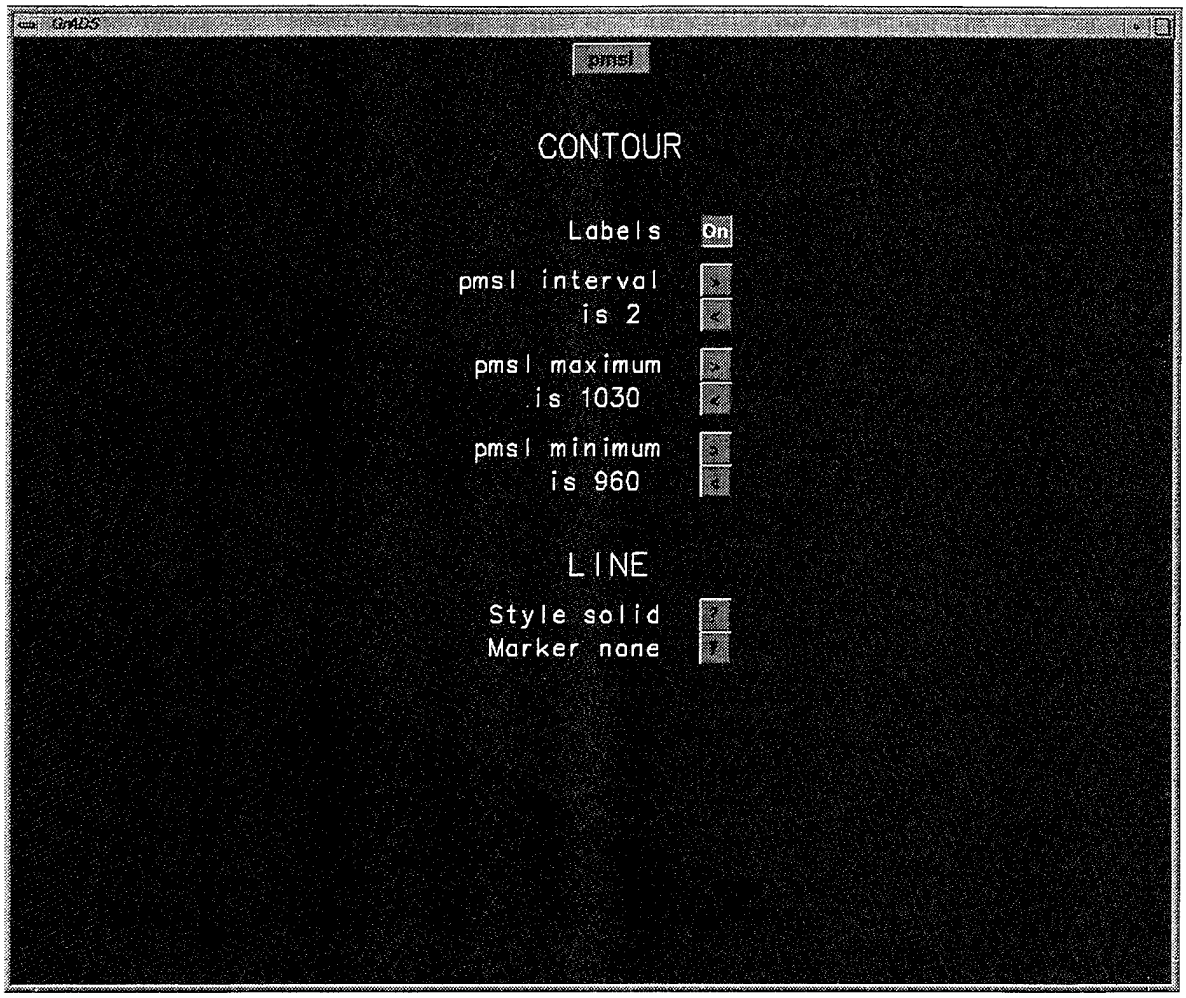


Figure 3. Variable customization menu.

4.3.3 Contour maximum. The maximum value contoured (or shaded) can be incremented or decremented by clicking the '>' or '<' symbol, respectively.

4.3.4 Contour minimum. The minimum value contoured (or shaded) can be incremented or decremented by clicking the '>' or '<' symbol, respectively.

4.3.5 Line style. The style of line used for plotting can be selected as 'solid', 'long dash', 'short dash', or 'dotted' by clicking on the '?' symbol until the desired option appears.

4.3.6 Line marker. The marker for plotting data can be selected as 'none', 'cross', 'open circle', 'closed circle', 'open square', or 'closed square' by clicking on the '?' symbol until the desired option appears.

4.3.7 Skip interval. Vector variables can be decimated by skipping grid points. When skip is set to '0' every point is plotted. If skip is set to '1', '2', or 'n', the 2nd, 3rd, or n+1th points in latitude and longitude are plotted.

4.4 Graph Type Selection. By default, the graph type that is *highlighted when a variable is selected* will be used to plot that variable.

4.4.1 Contour. This graph type is the most common for scalar variables and can be for as many variables as desired. Each subsequent usage of contouring within the same graph will produce a different contour color and corresponding label color.

4.4.2 Shaded. The 'shaded' graph type can only be used once for a scalar variable since a second selection would overdraw the first plotted variable. If selected a second time, a warning is issued and the name of the first shaded variable is given so its graph type may be changed or the variable may be deselected.

4.4.3 Vector. This graph type places vectors with their tail at their origin plotted in the direction the wind is going. It is primarily for wind data and is similar to the shaded graph type in that it can only be used for one pair of vector variables.

4.4.4 Barb. This graph type places barbs with their head at their origin plotted in the direction from where the wind is coming. It is primarily for wind data and is similar to the shaded graph type in that it can only be used for one pair of vector variables.

4.4.5 Stream. This graph type produces magnitude colored streamlines and is primarily for wind data and is similar to the shaded graph type in that it can only be used for one pair of vector variables.

4.4.6 Mixed. Contours of isotachs can be overlaid on vector, barb, or streamline displays of winds by selecting one wind component (i.e., u or v) with 'contour' and selecting the other wind component with 'vector', 'barb' or 'stream'.

4.5 Date-Time Selection. The time span contained within the data set is displayed as a vertical bar with a tick mark for each time group in the data. Either data set time marks or interpolated time marks can be selected.

4.5.1 Single time. A single time value is selected by clicking with the left mouse button on the desired time tick. The exact time and date of the selected tick is displayed.

4.5.2 Time range. The first time value is selected using either the left or middle mouse button. Then the second time value is selected with the middle mouse button. The selected range is highlighted and the beginning and ending date-time values are displayed.

4.5.3 Time settings. The current time interval between tick marks is displayed vertically at the center of the time bar. Clicking with the right mouse button on the time interval displays the time settings menu (Fig. 4). Options for displaying a local clock or setting the desired time interpolation interval are available. Options must be selected using the right mouse button. Clicking the left or middle mouse button accepts the displayed time option settings.

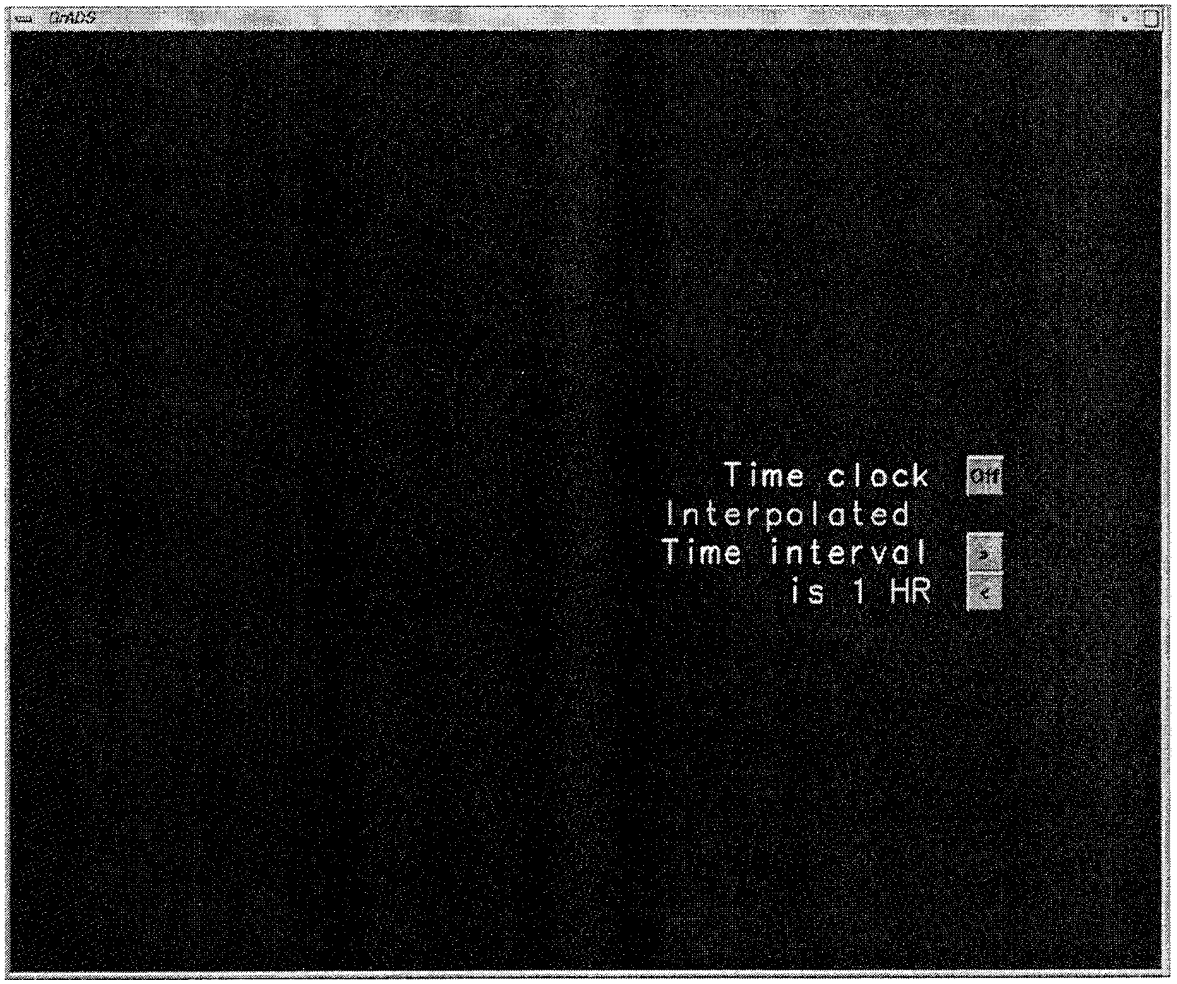


Figure 4. Time settings menu.

4.5.3.1 Local time clock. A clock face showing the local time can be toggled on or off by clicking its On/Off status button. The local time offset is the hour calculated using the display's center longitude rounded to the nearest 15° increment from GMT.

4.5.3.2 Time interpolation. The time interval can be increased or decreased by clicking the '>' or '<' symbol, respectively. If an interpolated interval is selected the vertical time bar indicates the status (Fig. 5) and the interpolated time intervals are displayed with yellow tick markers on the time bar.

4.6 Dimension Selection. The default spatial dimension is level (Z) and may be quoted in feet, meters, sigma levels, etc., as determined by the data set. Longitude (X) or latitude (Y) in degrees may be selected by clicking on the appropriate button using any mouse button. The range and selected dimensional values are displayed on a vertical bar similar to time.

4.7 Level Selection. Select 'Lev' using any mouse button. The vertical extent of the data set is displayed as a vertical bar with a tick mark for each available vertical level.

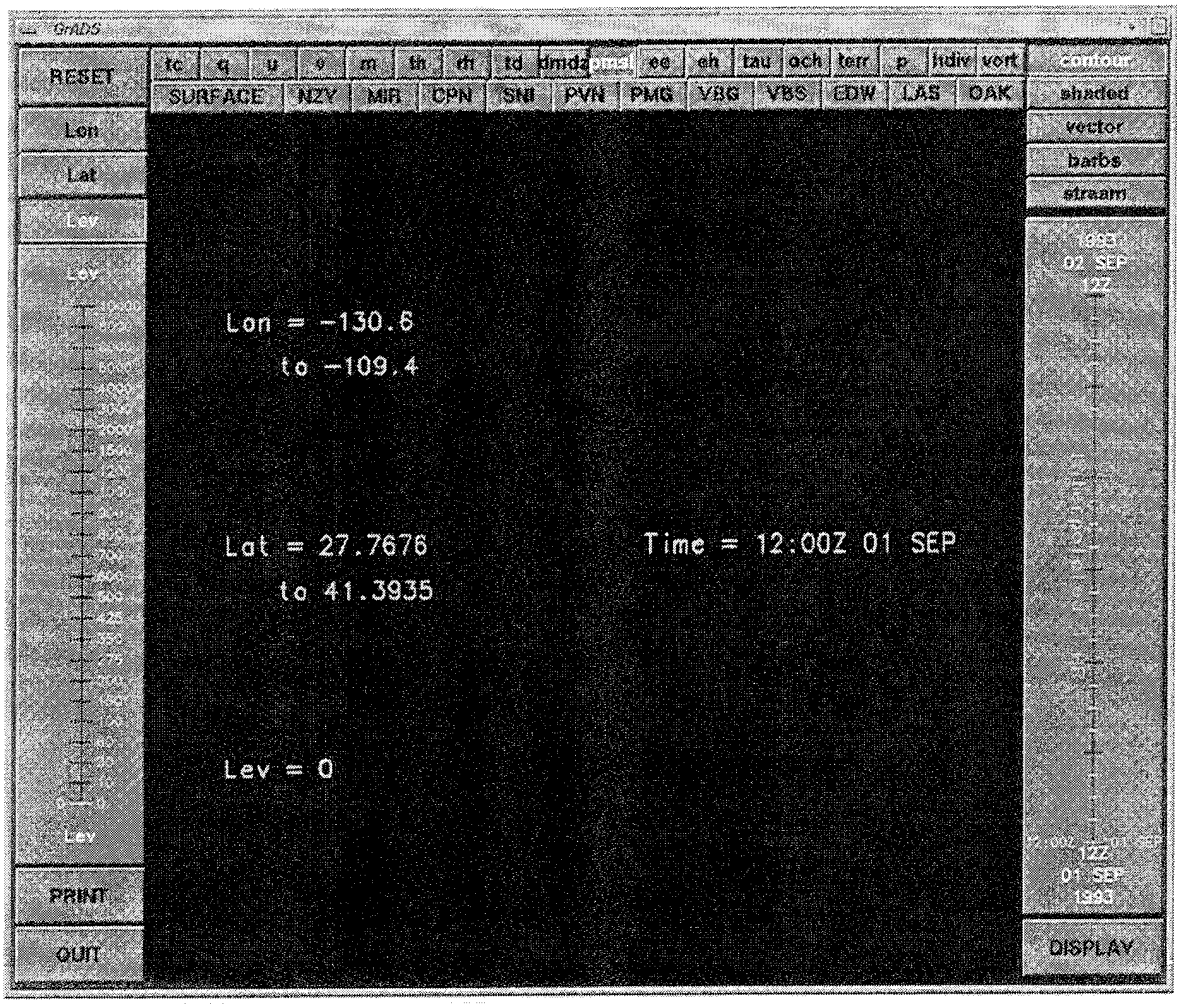


Figure 5. Customization of time interval.

4.7.1 Single level. A single level value is selected by clicking with the left mouse button on the desired level's tick mark. The exact level of the selected tick is displayed.

4.7.2 Level range. The first level value is selected using either the left or middle mouse button. Then the second level value is selected using the middle mouse button. The selected range is highlighted and the beginning and ending level values are displayed.

4.8 Latitude Selection. Select 'Lat' using any mouse button. The north-south extent of the data set is displayed as a vertical bar with a tick mark for each available latitude.

4.8.1 Single latitude. A single latitude value is selected by clicking with the left mouse button on the desired latitude's tick mark. The exact latitude of the selected tick is displayed.

4.8.2 Latitude range. The first latitude value is selected using either the left or middle mouse button. Then the second latitude value is selected with the middle mouse button. The selected range is highlighted and the beginning and ending latitude values are displayed.

4.9 Longitude Selection. Select 'Lon' using any mouse button. The east-west extent of the data set is displayed as a vertical bar with a tick mark for each available longitude.

4.9.1 Single longitude. A single longitude value is selected by clicking with the left mouse button on the desired longitude's tick mark. The exact longitude of the selected tick is displayed.

4.9.2 Longitude range. The first longitude value is selected using either the left or middle mouse button. Then the second longitude value is selected with the middle mouse button. The selected range is highlighted and the beginning and ending longitude values are displayed.

4.10 Display and Animation. The selected graph types, variables and conditions are displayed by clicking on 'DISPLAY' with any mouse button, or by clicking the right mouse button in the central clear area of the window. (Note: Clicking on a dimension bar or time bar with the right mouse button will select a new single dimension or time value and immediately re-display the selected variables and graph types.) Display and animation actions are inferred from the fixed level, or varying range status of the time bar and the indicated dimension bar.

4.10.1 Single frame. In general, if a single fixed time is selected and the indicated dimension has a single fixed value selected, only a single frame will be displayed. Exceptions are shown in Table 1 and will be discussed in Section 5.

4.10.2 Animation. In general, if the time bar shows a varying range selected, an animation in time will result. In general, if the indicated dimension bar shows a varying range is selected, an animation in that dimension will result. Exceptions are shown in Table 1 and will be discussed in Section 5.

Table 1. Gridded data display options.

LON (X)	LAT (Y)	LEV (Z)	TIME (T)	DIMENSION	RESULT
Fixed	Fixed	Fixed	Fixed	Any	Printed values
Fixed	Fixed	Fixed	Varying	Any	Time line
Varying	Fixed	Fixed	Fixed	Any	Line plot vs. X
Varying	Fixed	Fixed	Varying	X Y Z	X-T section T animated line plot vs. X T animated line plot vs. X
Fixed	Varying	Fixed	Fixed	Any	Line plot vs. Y
Fixed	Varying	Fixed	Varying	X Y Z	T animated line plot vs. Y Y-T section T animated line plot vs. Y
Varying	Varying	Fixed	Fixed	X Y Z	Line plot vs. Y animated in X Line plot vs. X animated in Y X-Y map
Varying	Varying	Fixed	Varying	X Y Z	Y-T section animated in X X-T section animated in Y X-Y map animated in T
Fixed	Fixed	Varying	Fixed	Any	Z vs. variable
Fixed	Fixed	Varying	Varying	X Y Z	T animated Z vs. variable T animated Z vs. variable Z-T section
Varying	Fixed	Varying	Fixed	X Y Z	Z vs. variable animated in X Z-X section Line vs. X animated in Z
Varying	Fixed	Varying	Varying	X Y Z	Z-T section animated in X Z-X section animated in T X-T section animated in Z
Fixed	Varying	Varying	Fixed	X Y Z	Z-Y section Z vs. variable animated in Y Line vs. Y animated in Z
Fixed	Varying	Varying	Varying	X Y Z	Z-Y section animated in T Z-T section animated in Y Y-T section animated in Z
Varying	Varying	Varying	Fixed	X Y Z	Z-Y section animated in X Z-X section animated in Y X-Y map animated in Z
Varying	Varying	Varying	Varying	Any	Prohibited

5. **Plotting Gridded Data.**

5.1 Plot Types. Various combinations of fixed and varying conditions for latitude, longitude, level, and time can be selected. These combinations further interact with whether the indicated dimension is fixed or varying. The resulting plot types for the combinations shown in Table 1 are discussed below:

5.2 No Plot.

- All dimensions and time are fixed. The values of the selected dimensions, time, and variables are printed (Fig. 6).
- All dimensions and time are varying. This case is prohibited.

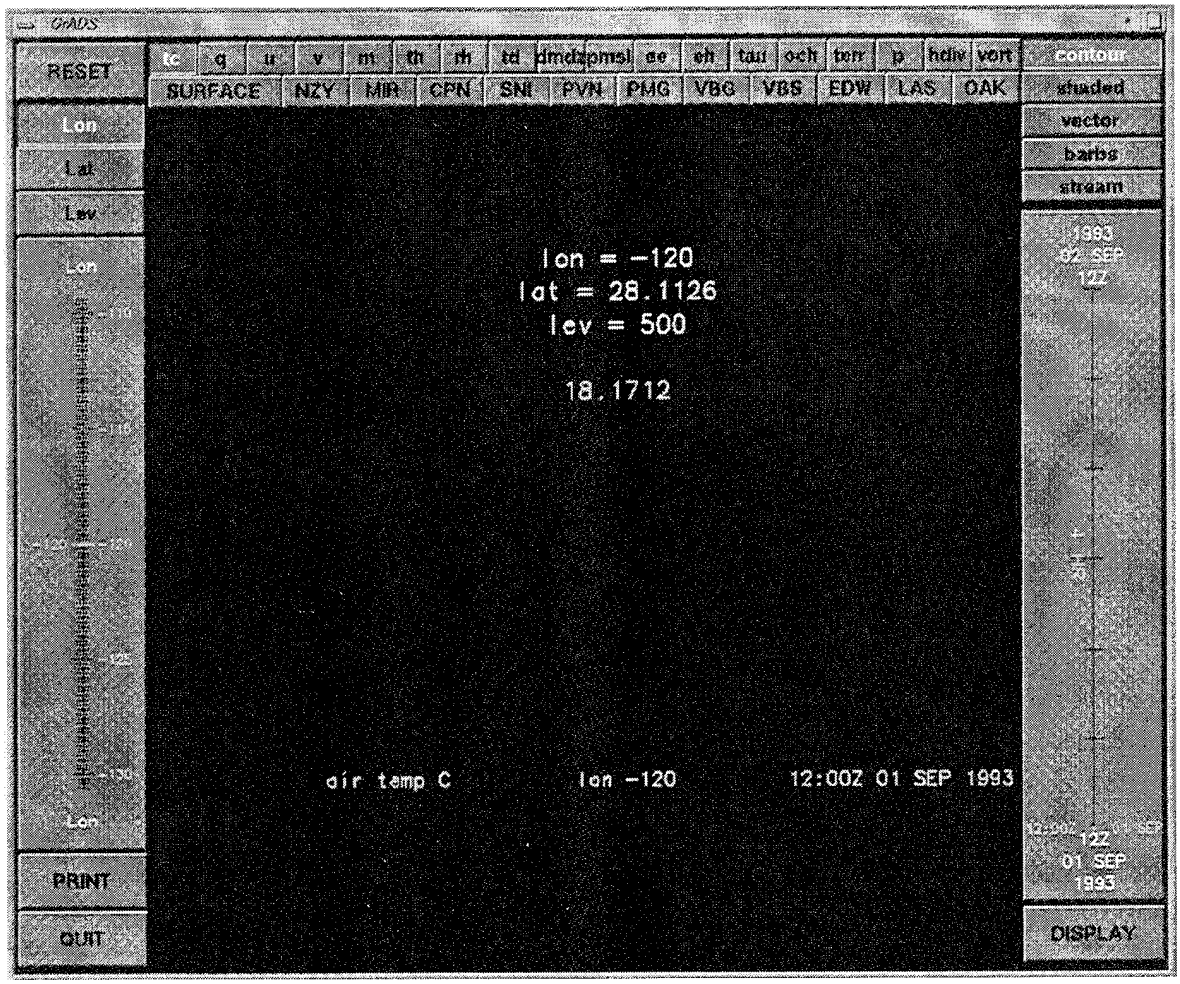


Figure 6. Printed data for fixed conditions.

5.3 Line Plot.

- One dimension or time is varying. Line plots of the selected variables are displayed in different colors with color coded axes and labels (Fig. 7).

- Time and a hidden dimension are varying. A time animation of a line plot for the hidden dimension will occur.

- The indicated dimension and a hidden dimension are varying. A slice animation of a line plot for the hidden dimension will occur with the indicated dimension varying.

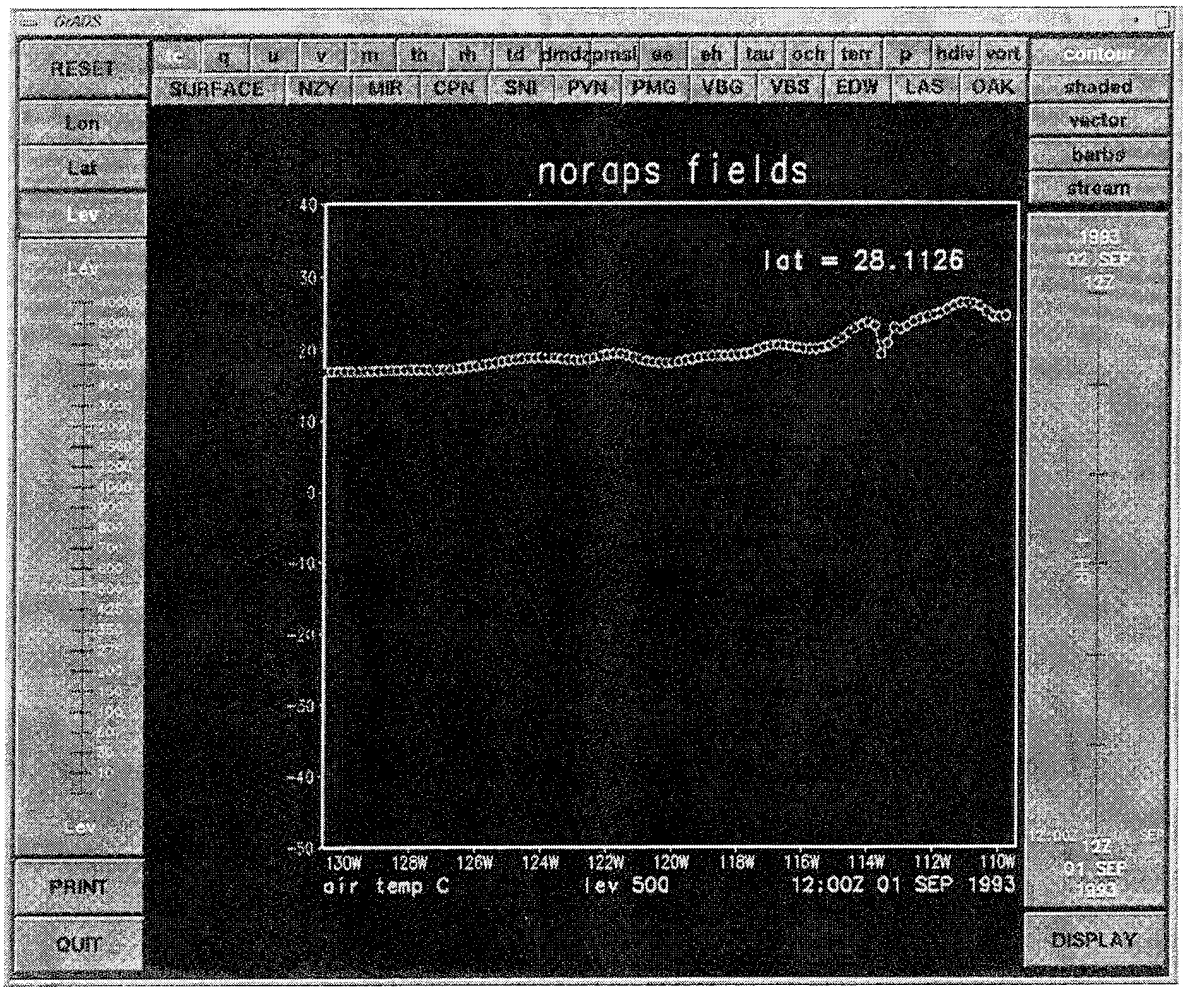


Figure 7. Typical line plot.

5.4 2D Section.

- One dimension and time are varying (Fig. 8). A static time cross section will occur for the indicated dimension.

- Two dimensions are varying, but time is fixed (Fig. 2). Map plots with coast lines and political boundaries are displayed if level is fixed. Vertical cross sections are displayed along the selected latitude (longitude), if latitude (longitude) is fixed (Fig. 9).

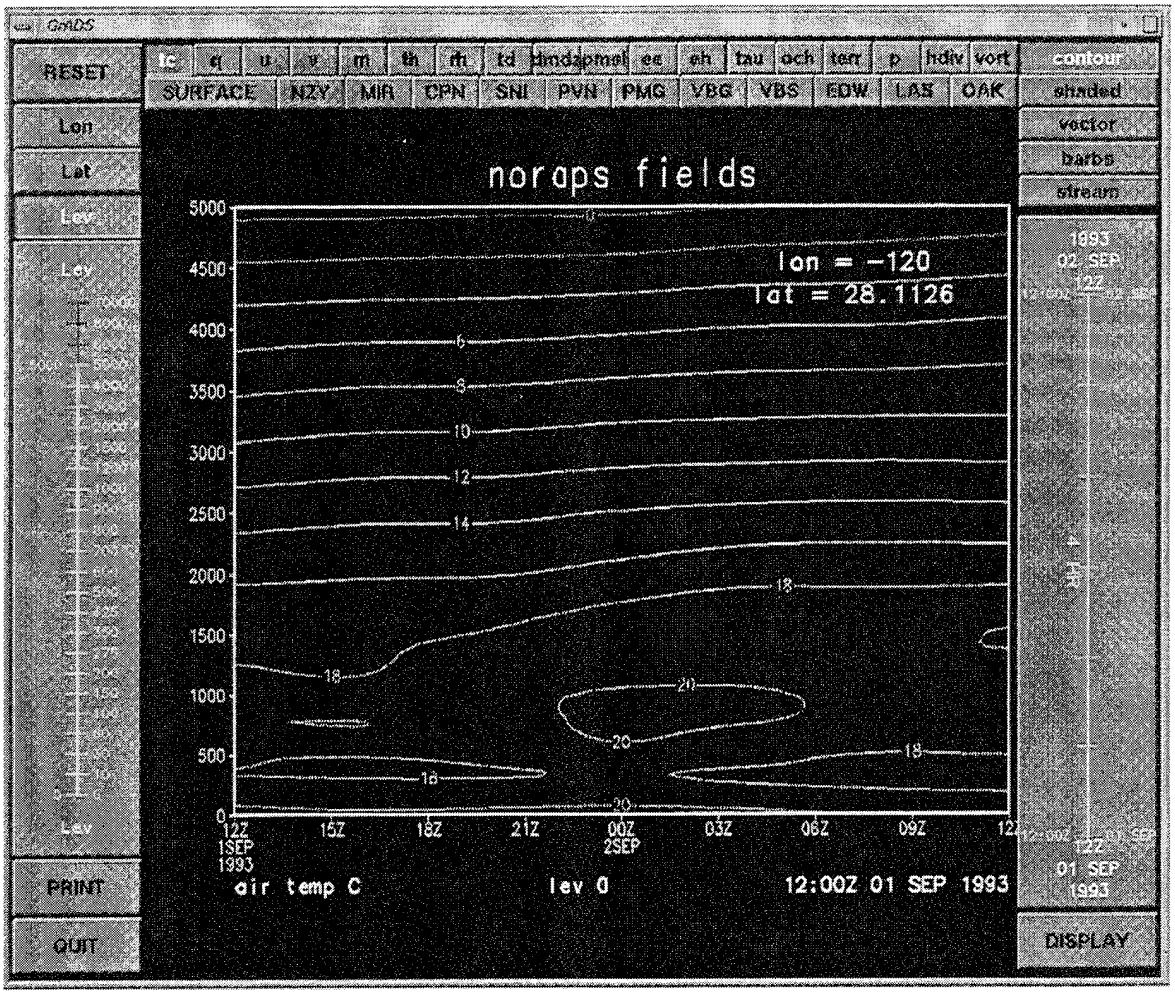


Figure 8. Typical Z-T time section.

- Two dimensions are varying and time is varying. A time animation of a map or cross section will occur if the indicated dimension is fixed, or time section will animate in the indicated dimension if it is varying.

- Three dimensions are varying. A slice animation of the maps will occur if level is the indicated dimension, or a slice animation of vertical cross sections will occur for latitude (longitude) if latitude (longitude) is the indicated dimension.

6. **Plotting Surface Data.**

6.1 Surface Station Availability. If surface station data are available for comparison with model surface forecasts, a 'SURFACE' button will appear under the row of variable buttons. Clicking on 'SURFACE' with any mouse button will toggle the 'SURFACE' button on and off.

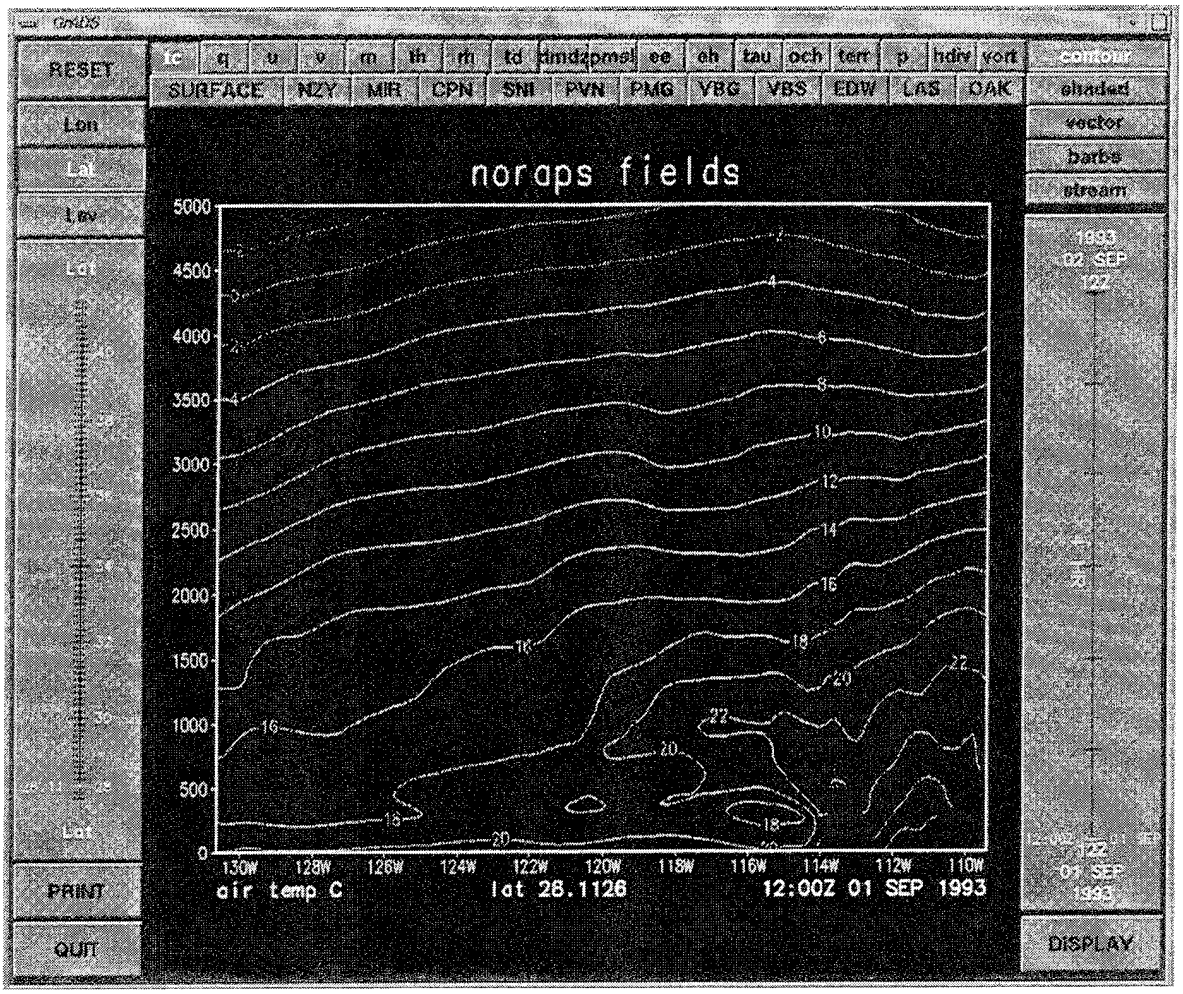


Figure 9. Typical Z-X vertical cross section

6.2 Surface Station Wind Display. Surface station wind data are displayed when a map display is selected (i.e., latitude and longitude varying with level and/or time fixed). If a time range is selected, the selected variable and the surface wind barbs will animate. If a time interpolation interval is selected, both the model data and the surface wind data will be interpolated and displayed. If a range of levels is selected, the model data will animate in Z, but the surface winds will remain fixed.

6.3 Surface Station Selection. Surface stations can be selected only from a fixed X-Y map display, without any animation in Z or T because the selection and animation processes are incompatible. The X-Y map display (Fig. 10) of any variable will be overlaid with surface wind barbs for all available surface stations. When station markers are visible, individual stations can be selected by clicking on the desired station's marker using the middle mouse button. The station will be highlighted and its 3-character nomenclature will be displayed to its upper right. Repeated clicking on a station with the middle mouse will toggle its selection on and off.

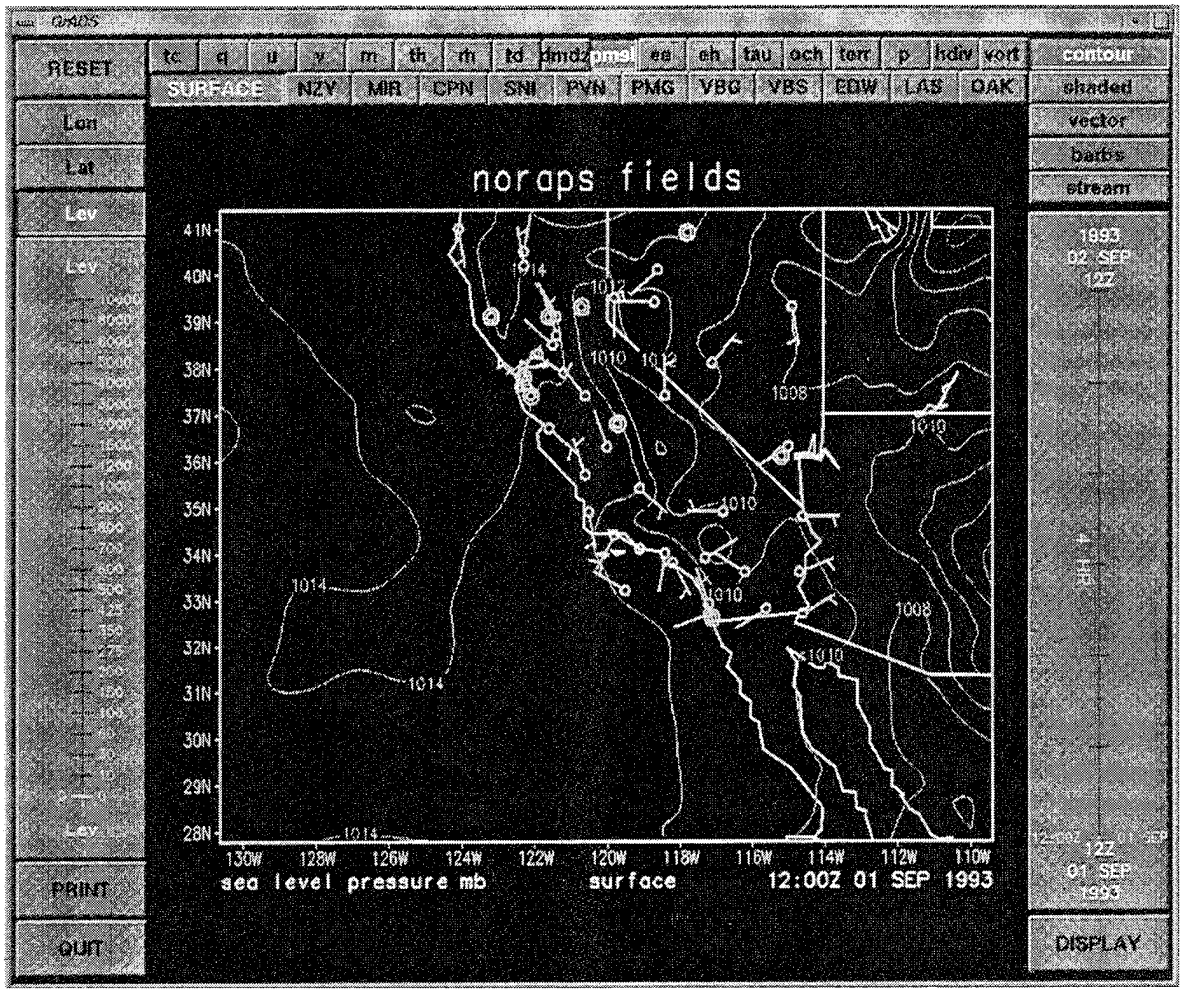


Figure 10. Display for surface station selection.

6.4 Surface Time Line Display. When the desired surface stations have been selected, clicking the middle mouse button outside the X-Y map display will cause the display of stacked time lines, one for each selected surface station (Fig. 11). Variable selections can be changed and new station time lines can be displayed by clicking on new variables followed by an immediate click with the middle mouse button in the central clear area of the window.

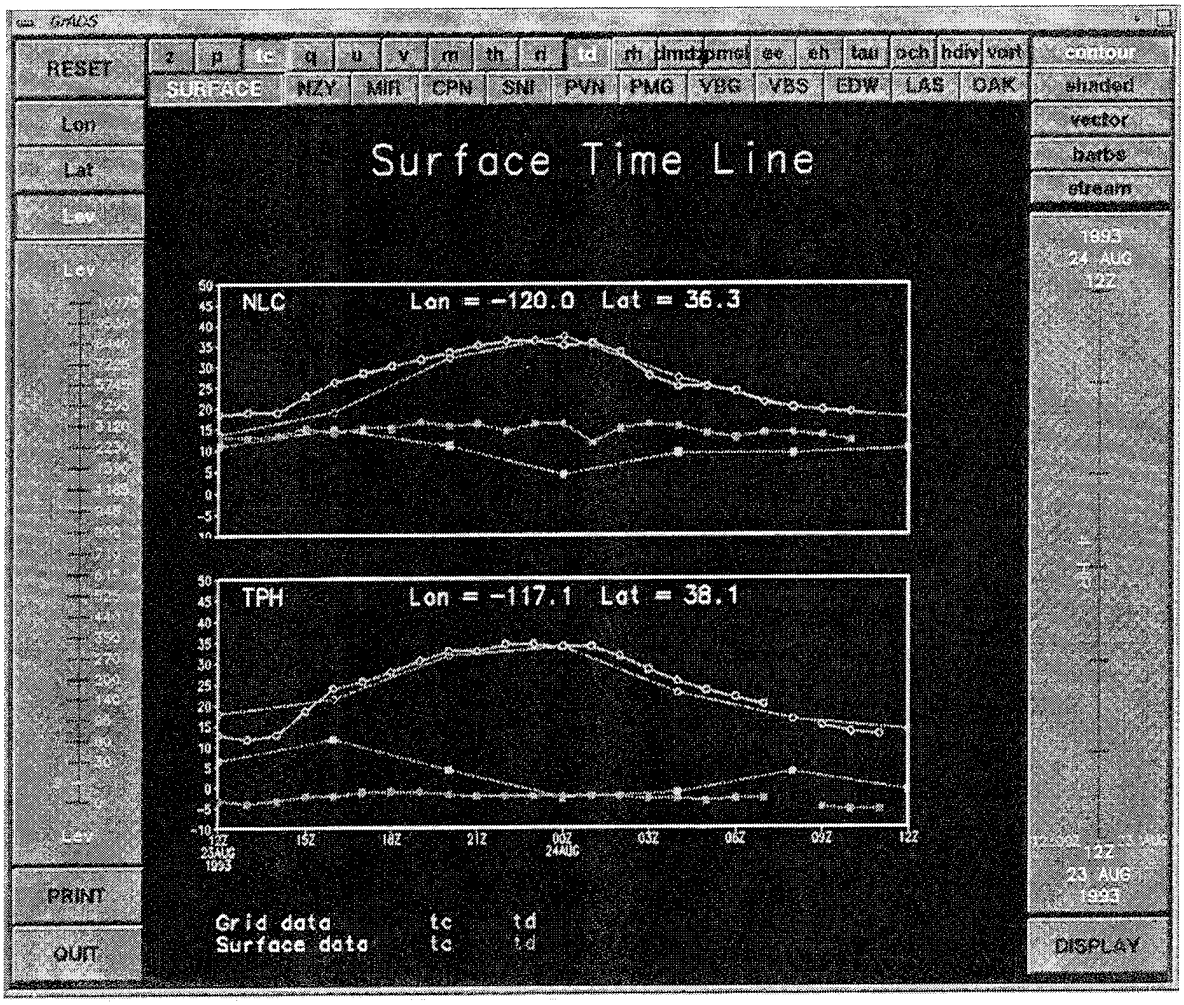


Figure 11. Typical surface time lines.

6.5 Model Time Section Display. Once stations have been selected, it is possible to plot a Z-T section of model data at the selected location by first defining the desired range in Z by using the middle mouse button on the level bar and then clicking in the center of the window with the middle mouse button (Fig. 12). (Note: To change the station selection you must first reset the level to a single value of Z or the display will animate in Z when the map display is activated.)

6.6 Surface Display Customization. The contour limit selections available under the variable menus can be changed to control the plot range of the selected variables. If no time range is selected the default is the full time range. Selecting a time range will expand the display in time to the desired interval. Model time section variables can be shaded or displayed as vectors.

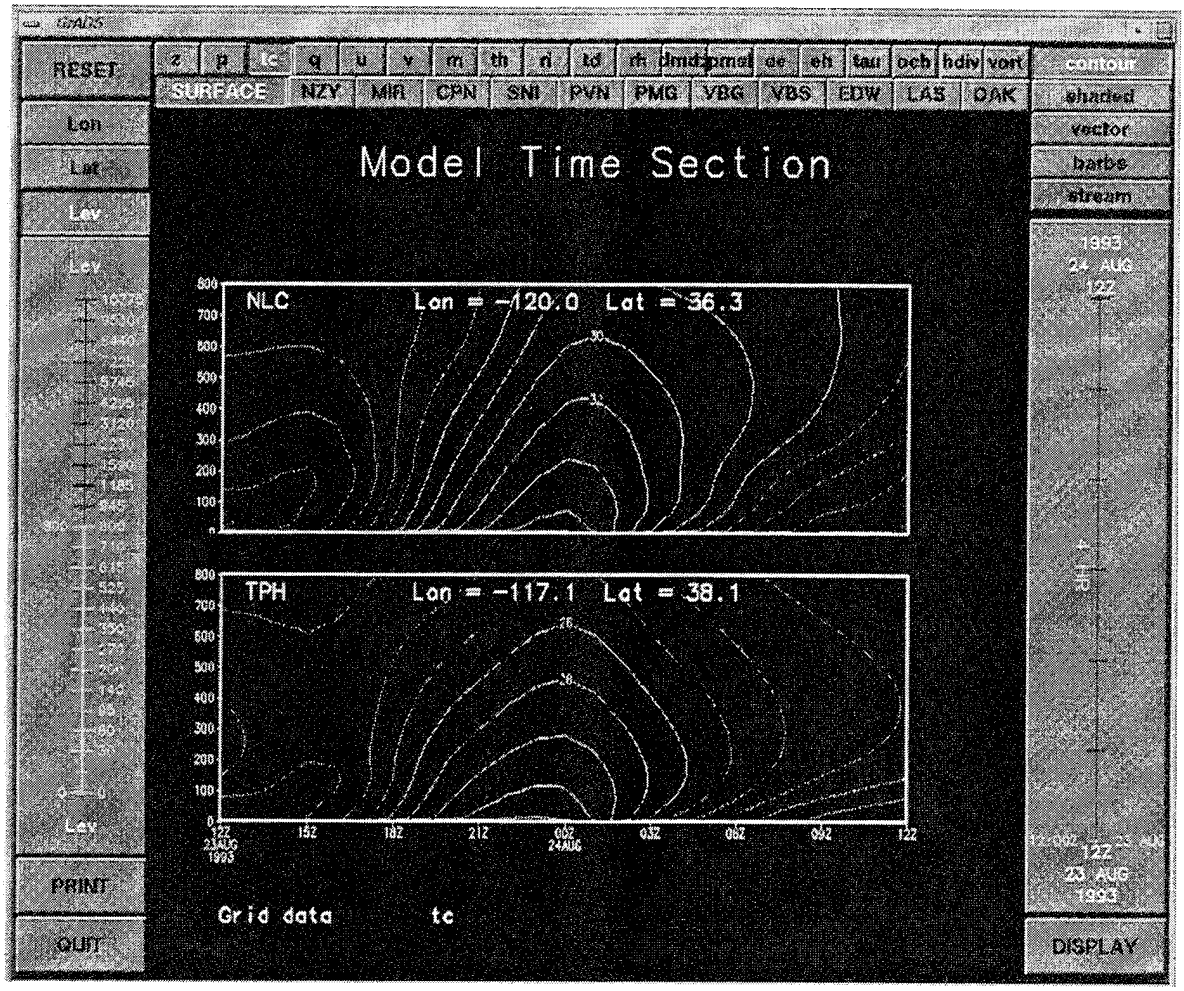


Figure 12. Typical model time section.

7. **Plotting Upper Air Data.**

7.1 Upper Air Station Availability. If upper air station data are available for comparison with model profiles, a row of station buttons will appear under the row of variable buttons with the 3-character nomenclature of each station indicated.

7.2 Upper Air Station Selection. Clicking on a station's button with any mouse button will toggle that station on and off. Since upper air data can be displayed for only one station, any previously selected stations will be deselected.

7.3 Upper Air Data Selection. Variables are selected the same as gridded and surface data by using the left or middle mouse button to toggle the variable selection.

7.4 Upper Air Station Data Display. Upper air data and model vertical profiles are displayed (Fig. 13) the same as gridded data by either clicking 'DISPLAY" or clicking the right mouse button in the window's central area.

7.5 Upper Air Station Data Animation. If a time range is selected, the model profiles will animate, but the upper air data will only appear when present at a given time step. If the time interval is interpolated, the same plotting behavior occurs.

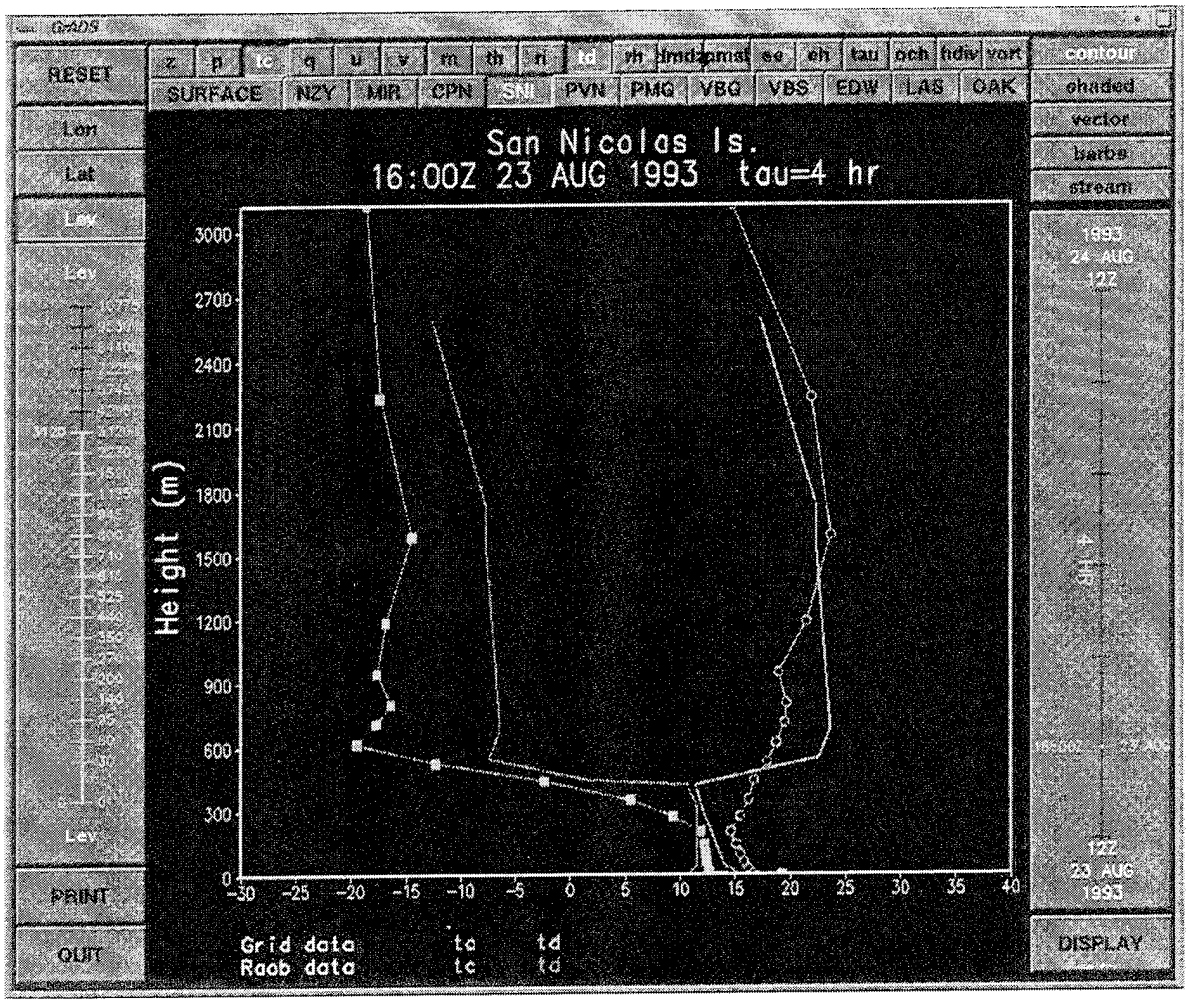


Figure 13. Typical radiosonde and model profiles.

8. Zooming and Panning Map Displays.

Only X-Y map displays of a gridded data variable can be zoomed in, zoomed out, and panned.

8.1 Zooming In. The first click within a map display using the left mouse button will cause the scale to be halved and the display to be re-centered on the location of the mouse cursor. A subsequent left mouse button click will zoom in only if near the map center. Each zoom results in a halving of the map scale, or a size doubling of map features.

8.2 Panning. If the left mouse button click is outside the central 20% of the map, the scale will remain the same, but the map will be re-centered on the new mouse cursor location. This allows the ability to pan without further zooming until the feature of interest is near the center, when zooming in can be continued if desired.

8.3 Zooming Out. The assigned plot area has a hidden boundary that is sensitive to clicks with the left mouse button. Generally, a left mouse button click on or outside the map border will result in a doubling of the map scale. The area under the map has a larger sensitive band that provides a more reliable zoom out activation.

9. **Printing Displays and Animation Sequences.**

Executing any print option re-displays the last display request and sends it to the selected destination. Clicking with any mouse button on 'PRINT' will display the print menu (Fig. 14). Clicking on a selection will activate that choice. A large number of frames can be easily requested which may overload the selected printer and local disk storage capacity. The number of frames is quoted and a final choice to continue is required. The print function is implemented in a companion script called 'output.gs' which must be edited to include print commands appropriate to your system.

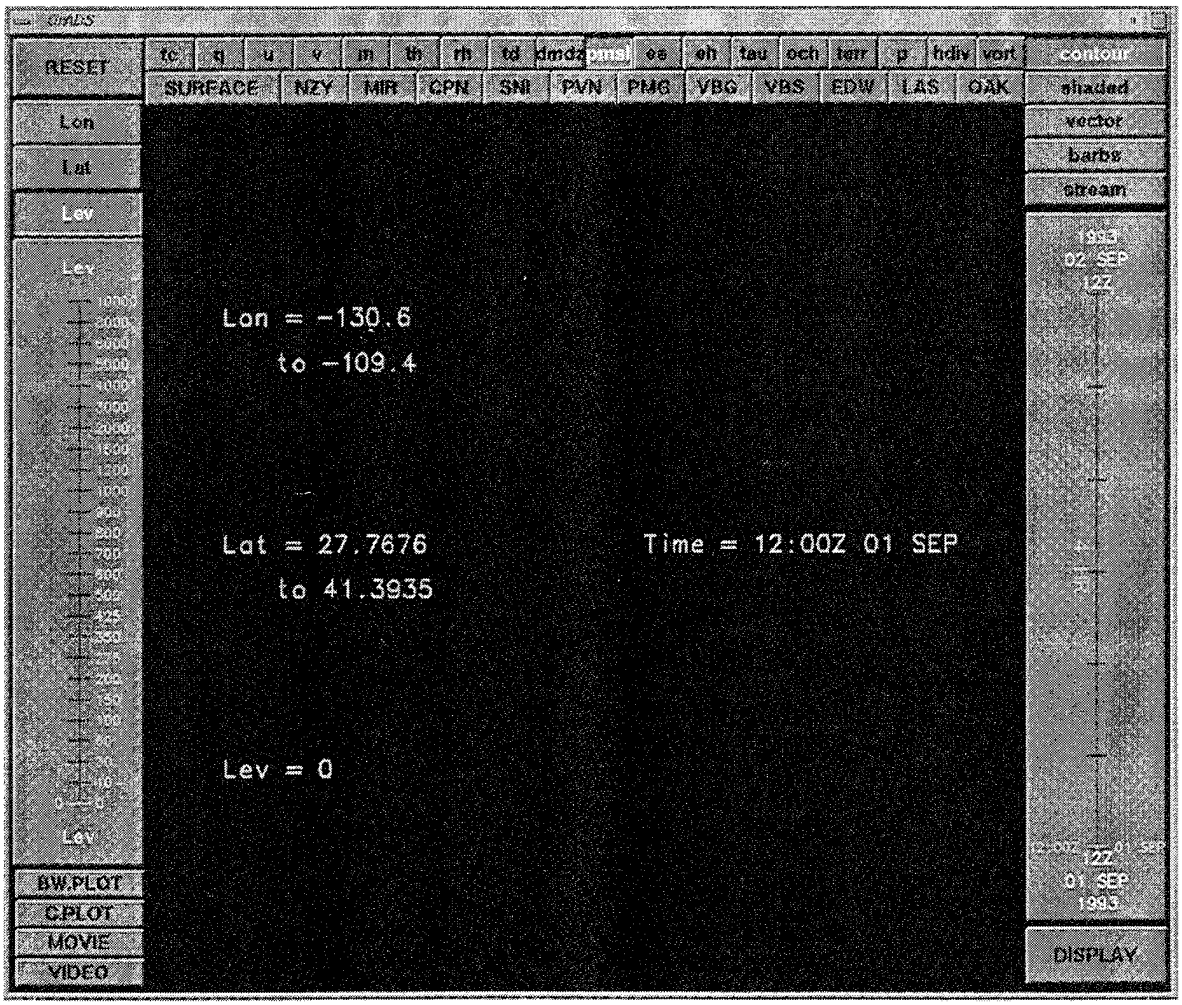


Figure 14. Print menu.

9.1 BW.PLOT. The requested frames are plotted to a local black and white printer, or a network printer, as defined in the 'output.gs' script.

9.2 C.PLOT. The requested frames are plotted to a network color printer, as defined in the 'output.gs' script.

9.3 MOVIE. The requested frames are plotted in a frame animator on the host display. This allows a quick preview of the requested frames before they are printed or sent to the video editor.

9.4 VIDEO. The requested frames are properly reformatted and sent to a commercial video editor where title frames can be added and the frames can be recorded to VHS video tape. (This currently is not implemented. The necessary commands and data paths must be set up in the companion GrADS script 'output.gs'.)

10. Resetting.

Occasionally, it may be convenient to return to the initial settings that existed when NEBS was started. Rather than resetting many conditions individually, any mouse button click on the 'RESET' button will re-initialize all variables and conditions.

11. Exiting.

The NEBS session should be terminated by a mouse button click on the 'QUIT' button. This returns control to the GrADS prompt in the originating xterm window.

12. Summary.

The NRL Environmental data Browsing System (NEBS) is able to browse any data set provided that it is in a GrADS compatible data set format. NEBS is capable of easily producing animations in time or a selected dimension, x, y, or z. These animations allow a qualitative assessment of data consistency and model behavior and easily reveal anomalies and erroneous trends. By setting time and spatial dimensions at the location of an identified problem, NEBS gives a quantitative listing of individual variables. Variables and dimensions can be easily set using mouse button clicks on the displayed GUI buttons. Graphics are displayed quickly and can be animated on most host platforms at about one frame per second. Over a high speed LAN this degrades to about one frame per two seconds. Selected data displays can be easily printed, in black & white or color, or converted to on-screen animations or video VHS tape.

13. Recommendations.

GrADS provides excellent data handling and 2-D graphical display of gridded data. However, upgrading GrADS or coupling GrADS to a 3-D graphics system, such as VIS-5D, is highly desirable to provide 3-D volume renderings of isosurfaces and wind fields. NEBS should be revised whenever a 3-D rendering capability becomes available for GrADS.

ACKNOWLEDGEMENTS

The original concept for the Naval Environmental data Browsing System was inspired by the GrADS data browser created by Steve J. Lord at the National Meteorological Center. The GrADS author, Brian Doty, and other GrADS users, especially, Dr. Michael Fiorino were very helpful in the NEBS development. Also Dr. Stephen Burk and Dr. William Thompson of NRL are thanked for their guidance and feedback during NEBS development. The support of the sponsor, the Space and Naval Warfare Systems Command under program element 0603207N, is gratefully acknowledged.

REFERENCES

Doty, B. and J. Kinter, 1992: The Grid Analysis and Display System (GrADS): A practical tool for earth science visualization, *Eighth International Conference on Interactive Processing Systems for Meteorology, Oceanography and Hydrology,* Jan 1992, Atlanta, GA (Amer. Meteor. Soc.), 115-116.

Doty, B. and J. Kinter, 1993: The Grid Analysis and Display System (GrADS): An Update, *Ninth International Conference on Interactive Processing Systems for Meteorology, Oceanography and Hydrology,* Jan 1993, Anaheim, CA (Amer. Meteor. Soc.), 165-167.

This Page Intentionally Left Blank.

Appendix A: Data Input Formats

This Page Intentionally Left Blank.

GrADS Data Sets[1]

A.1. Gridded Data Sets.

The GrADS gridded data set is a direct access binary data set. It may contain any number of variables at specified longitude, latitude, vertical, and time intervals.

GrADS views this data set as a giant array -- with X (longitude) varying the fastest, then Y (latitude), then Z (vertical level), then the variable type, then T (time).

It is easier for us to think of the data set in terms of a sequence of horizontal grids, where longitude and latitude vary. Each horizontal grid represents a particular variable at a particular height and time. Each horizontal grid is the same size in any particular GrADS data set (if you have grids of different sizes, you must create separate data sets).

These grids are written to the data set in the following order: starting with a particular variable, grids for each vertical level (at a particular time) are written out in ascending order. Then the grids for the next variable are written out. When all the grids at a particular time have been written, grids for the next time are written.

The format of this data set is thus exactly the same as the COLA Pressure History format, except: there are no date/time records, and latitude varies from south to north (not north to south as in the pressure history data).

Each binary gridded data set is described by a data descriptor file, essentially a table of contents for the binary data set. Following is an example of such a file:

```
DSET    ua.dat
TITLE Upper Air Data
UNDEF  -9.99E33
XDEF    80 LINEAR  -140.0  1.0
YDEF    50 LINEAR   20.0   1.0
ZDEF   10 LEVELS  1000 850 700 500 400 300 250 200 150 100
TDEF    4 LINEAR 0Z10apr1991  12hr
VARS    5
slp     0   0  sea level pressure
z      10   0  heights
t      10   0  temps
td      6   0  dewpoints
u      10   0  u winds
v      10   0  v winds
ENDVARS
```

---

[1] Extracted from GrADS documentation with permission from Brian Doty, Center for Ocean-Land-Atmosphere Interactions (COLA), Department of Meteorology, University of Maryland.

The data descriptor file is 'free format', i.e. each entry is blank delimited and may appear in any column. Comment records start with an asterisk ('*') in column 1. Comments may not appear in the list of variable records (between the vars and endvars records). Records may not be more than 80 characters long.

In this example, the binary data set is named ua.dat, the undefined, or missing, data value is -9.99e33, there are 80 grid points in the X direction, 50 in the Y direction, 10 levels, 4 times, and 5 variables. The variables z, t, u, and v have 10 levels, the variable td has 6 levels, and the variable slp has one level (see below for a more specific description of each entry).

Think in terms of the X and Y data points at one level for one variable at one time being a horizontal grid. This grid is exactly in the same storage order as a FORTRAN array, in this case an array DIMENSION A(80,50). The first dimension always varies from west to east, the second from south to north.

In the above example the horizontal grids would be written in the following order:

Time 1, Level ?, Variable slp
Time 1, Level 1000, Variable z
Time 1, Level 850, Variable z
then levels 700, 500, 400, 300, 250, 200, then
Time 1, Level 150, Variable z
Time 1, Level 100, Variable z
Time 1, Level 1000, Variable t
Time 1, Level 850, Variable t
then levels 700, 500, 400, 300, 250, 200, then
Time 1, Level 150, Variable t
Time 1, Level 100, Variable t
Time 1, Level 1000, Variable td
Time 1, Level 850, Variable td
Time 1, Level 700, Variable td
Time 1, Level 500, Variable td
Time 1, Level 400, Variable td
Time 1, Level 300, Variable td
Time 1, Level 1000, Variable u
Time 1, Level 850, Variable u
then levels 700, 500, 400, 300, 250, 200, then
Time 1, Level 150, Variable u
Time 1, Level 100, Variable u
Time 1, Level 1000, Variable v
Time 1, Level 850, Variable v
then levels 700, 500, 400, 300, 250, 200, then
Time 1, Level 150, Variable v
Time 1, Level 100, Variable v
Time 2, Level ?, Variable slp
Time 2, Level 1000, Variable z

Time 2, Level 850, Variable z
Time 2, Level 700, Variable z
Time 2, Level 500, Variable z
Time 2, Level 400, Variable z
.
.
.
etc

A description of each record in the GrADS data descriptor file follows:

DSET data-set-name

This entry specifies the name of the binary data set. It may be entered in mixed case.

If the binary data set is in the same directory as the data descriptor file, you may enter the filename in the data descriptor file without a full path name by prefixing it with a ^ character. For example, if the data descriptor file is:

/data/wx/grads/sa.ctl

and the binary data file is:

/data/wx/grads/sa.dat

you could use the following file name in the data descriptor file:

DSET ^sa.dat

instead of:

DSET /data/wx/grads/sa.dat

As long as you keep the two files together, you may move them to any directory without changing the entries in the data descriptor file.

TITLE string

A brief description of the contents of the data set. This will be displayed during a QUERY command, so it is helpful to put meaningful information here.

UNDEF value

The undefined, or missing, data value. GrADS operations and graphics routines will ignore data with this value from this data set.

FILEHEADER bytenum

bytenum - the number of bytes of non-data at the beginning of the file

Indicates the binary data file has a header where bytenum is the number of bytes in the header. GrADS will skip past this header, then treat the file as though it were a normal GrADS file after that point. This option is valid only for GrADS gridded data sets.

OPTIONS <keywords>

Some keywords:
options <yrev> <zrev> <sequential> <template>
<big_endian> <little_endian>

yrev - y data in file goes from north to south
zrev - z data in file goes from top to bottom
sequential - FORTRAN unformatted sequential
template - file name templates in use
big_endian - data is big_endian
little_endian - data is little_endian

Sequential: The sequential option may be set if the data were written using unformatted (f77) I/O.

Template (for multiple file time series): GrADS allows you to handle many actual data files as one GrADS file, if the individual data files are in a GrADS readable format, and if the files are split along time. In the initial implementation, the time(s) that are in each file are indicated by the file name.

An example of this might be hourly data, where each 24 hours has been placed in a separate file. Each file is named this way:

1may92.dat
2may92.dat
etc.

You indicate to GrADS that there are multiple files in this time series by giving a substitution template as the file name:

dset %d1%mc%y2.dat

and giving an options record that looks like:

options template

and specifying the time range and increment in the tdef record:

tdef 72 linear 0z1may1993 1hr

GrADS will figure out automatically that there are 24 times in each file, and what file names correspond to what times. As you display data, GrADS will only open one file at a time. As you change times such that another file is referred to, the open file is closed, and the new file is opened.

Valid substitutions are:

%y2 - 2 digit year (last 2 digits)
%y4 - 4 digit year
%m1 - 1 or 2 digit month
%m2 - 2 digit month (leading zero if needed)
%mc - 3 character month abbreviation
%d1 - 1 or 2 digit day
%d2 - 2 digit day
%h1 - 1 or 2 digit hour
%h2 - 2 digit hour

This support works on all supported GrADS data types (GrADS gridded, GRIB, GrADS station data). If you specify file format options, the options must apply equally to each file.

Byte ordering: You may specify the actual byte ordering of the binary data file:

options big_endian

or

options little_endian

Indicates the binary data file is in reverse byte order from the normal byte order of the machine.

If the data is already in the correct order, no conversion is performed. If you then move the data set to another machine, conversion will be automatically performed. This would happen if you sent a file in binary format from, for example, a Sun to a PC. These options were added to facilitate moving data files and descriptor files between machines. Putting this keyword in the descriptor file tells GrADS to correct the byte order as the data is being read.

XDEF number <LINEAR start increment>
<LEVELS value-list>

Defines the mapping between grid values and longitude.

Specifically:
number -- the number of grid values in the X direction,
specified as an integer number.
Must be >= 1.
LINEAR or LEVELS -- Indicates the grid mapping type.

For LINEAR:

start -- the starting longitude, or the longitude for X = 1.
Specified as a floating point value, where negative
indicates degrees west.
increment -- the spacing between grid value in the X direction.
It is assumed that the X dimension values go from
west to east. Specified as a positive floating value.

For LEVELS:

value-list -- List of 'number' values representing the
longitude of each X dimension. May start and
continue on the next record in the descriptor
file (records may not be > 80 characters).
There must be at least 2 levels (otherwise
use LINEAR mapping).

YDEF number mapping start <increment>
<LEVELS value-list>

Defines the mapping between grid values and latitude.

Specifically:
number -- the number of grid values in the X direction,
specified as an integer number.
mapping -- mapping type, specified as a keyword.
Valid are:
LINEAR -- Linear mapping
GAUSR15 -- Gaussian R15 latitudes
GAUSR20 -- Gaussian R20 latitudes
GAUSR30 -- Gaussian R30 latitudes
GAUSR40 -- Gaussian R40 latitudes
start -- For LINEAR mapping, the starting latitude, ie the
latitude for Y = 1, and is specified as a floating

point value, with negative indicating degrees south. For GAUSRxx mapping, the start value indicates the first gaussian grid number, where 1 would be the southernmost gaussian grid latitude.

increment -- the spacing between grid values in the Y direction. It is assumed that the Y dimension values go from south to north. Specified as a positive floating point value. Used only for LINEAR mapping.

For LEVELS:

value-list -- List of 'number' values representing the latitude of each X dimension. May start and continue on the next record in the descriptor file (records may not be > 80 characters). There must be at least 2 levels (otherwise use LINEAR mapping).

Examples of specifying GAUSRxx mapping:

YDEF 20 GAUSR40 15

Indicates that there are 20 Y dimension values which start at Gaussian Latitude 15 (64.10 south) on the Gaussian R40 grid. Thus the 20 values would correspond to Latitudes:

-64.10, -62.34, -60.58, -58.83, -57.07, -55.32, -53.56,
-51.80, -50.05, -48.29, -46.54, -44.78, -43.02, -41.27,
-39.51, -37.76, -36.00, -34.24, -32.49, -30.73

YDEF 102 GAUSR40 1

The entire gaussian grid is present, starting at the southernmost latitude (-88.66).

ZDEF number mapping <start increment>
<value-list>

Defines the mapping between grid values and pressure level. Specifically:

number -- the number of grid values in the X direction, specified as an integer number.

mapping -- mapping type, specified as a keyword. Valid are:

LINEAR -- Linear mapping
LEVELS -- Arbitrary pressure levels

start -- when mapping is LINEAR, this is the starting value, or the value when Z=1.

increment -- when mapping is LINEAR, the increment in the Z direction, or from lower to higher. This may be a negative value, for example:

ZDEF 10 LINEAR 1000 -100

indicating that the data is for levels 1000, 900, 800, 700, etc.

value-list -- when the mapping is LEVELS, the specific levels are simply listed in ascending order. If there is only one level, use LINEAR, since LEVELS implies at least two levels.

TDEF number LINEAR start-time increment

Defines the mapping between grid values and time.

Specifically:

number -- the number of times in the data set. Specified as an integer number.

start-time -- The starting date/time value, specified in GrADS absolute date/time format. This is the value when T=1. The date/time format is:

hh:mmZddmmmyyyy

where:

hh = hour (two digit integer)
mm = minutes (two digit integer)
dd = day (one or two digit integer)
mmm = month (jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec)
yyyy = year (two or four digit integer. two digits implies a year between 1950 and 2049).

If not specified, hh defaults to 00, mm defaults to 00, and dd defaults to 1. The month and year must be specified. No intervening blanks are allowed in a GrADS absolute date/time.

Examples:

12Z1JAN1990
14:20Z22JAN1987
JUN1960

increment -- time increment. Specified in GrADS time increment format:

vvkk

where:

vv = an integer number, 1 or 2 digits
kk = an increment keyword,
mn = minutes
hr = hours
dy = days
mo = months
yr = year

Examples:

20mn -- increment is 20 minutes
1mo -- increment is 1 month
2dy -- increment is 2 days

Some examples of a TDEF statement:

TDEF 24 LINEAR 00Z01JUN1987 1HR

The data set has 24 times, starting at 00Z on 1 Jun, 1987, with an increment of 1 hour.

TDEF 30 LINEAR 2JUN1988 1DY

The data set has 30 times, starting at 00Z on 2 Jun, 1988, with an increment of 1 day.

VARS number

Indicates the start of the records describing the variables in the data set.

number -- the number of variable records

Each variable record is in the following format:

abrev levs units description

abrev -- a 1 to 8 character abbreviation for this variable. This abbreviation must start with an alphabetic character (a-z) and be composed of alphabetic characters and numbers. This abbreviation will be the "name" the variable is accessed by from within GrADS.

levs -- an integer value specifying the number of levels this variable has in the data set. It may not exceed the number of levels in the ZDEF statement. A levs value of 0 indicates this variable has one "level" that does not correspond to a vertical level. An example would be a surface variable.

units -- Reserved for future use. Put a value of 99 here.

description - A text description of the variable, max 40 characters.

After the last variable record comes the ENDVARS statement. This ends the GrADS data descriptor file.

GRIB data support: GrADS will now read some GRIB data files directly, particularly those being produced by NMC. The GRIB records must contain lat-lon grids. A detailed discussion on how to use GRIB data files is provided in GrADS documentation titled: update.151.

A.2. Station Data Sets.

Station data sets are written to a binary file one report at time. The only ordering required is that the station reports be grouped within the file into some time interval. For example, the time interval for upper air observations might be 12 hours.

Variables within each report are split into two groupings. Each variable is either a surface variable, thus can be reported at most once per report, or it is a level dependent variable, thus can be reported at a number of different levels within one report.

The format of a station report in the binary station data file is:

- A header which provides information about the location of the station.
- Surface variables, if any
- Level dependent variables, if any

The header is described by the following C language data structure:

```
struct rpthdr {
  char id[8];        /* Character station id          */
  float lat;         /* Latitude of report            */
  float lon;         /* Longitude of report           */
  float t;           /* Time in relative grid units   */
  int  nlev;         /* Number of levels following    */
  int flag;          /* Level independent var set flag */
};
```

A detailed description of each header entry follows:

id - The station identifier. This is a 1 to 7 character identifier that should identify the station uniquely. It may be assigned arbitrarily; ie. the stations could be numbered in some arbitrary order.

lat - The Y dimension location of the station in world coordinates, typically latitude.

lon - The X dimension location of the station in world coordinates, typically longitude.

t - The time of this report, in relative grid units. This refers to the way the stations are grouped in time. For example, if you are working with surface airways reports, you would probably have a time grouping interval of one hour. If you wanted to treat the report times of each report as being exactly on the hour, you would set t to 0.0. If the report was for 12:15pm, and you were writing the time group for 12pm, you would set t to be 0.25. Thus, t would typically have the range of -0.5 to 0.5.

nlev - Number of data groups following the header. This is the count of the one surface group, if present, plus the number of level dependent groups. Is set to zero to mark the end of a time group in the file.

flag - If zero, there are no surface variables following the header. If one, then there are surface variables following the header.

Following the header, the data for this report is written. The first group of data would be all the surface variables if present. Whether or not the surface variable (if any) are present is determined by the flag in the header. If present, then all the surface variables must be written -- missing variables should have the missing data value provided. Thus, each surface variable group will be the same size for each report in the file.

The surface variables are written out as floating point numbers. The ordering of the variables must be the same in each report, and is the ordering that will be given in the data descriptor file.

Following the surface variable group, any number of level dependent groups may be written. The number of total data groups is provided in the header. Each level dependent group must have all the level dependent variables present, even if they are filled with the missing data value. Thus, each level dependent group will be the same size for all levels and all reports in the file.

The level dependent group is written out as follows:

level -- floating point value giving the Z dimension
value in world coordinates for this level.
variables -- The level dependent variables for this level.

After all the reports for one time grouping have been written, a special header (with no data groups) is written to indicate the end of the time group. The header has an nlev value of zero. The next time group may then start immediately after. A time group with no reports would still contain the time group terminator header record (ie, two terminators in a row).

GrADS station data files must be written as UNIX stream data sets without any imbedded record descriptor information. This is easily done from a C program. From a FORTRAN program, it usually requires a system-dependent option in the OPEN statement. For example, in DEC FORTRAN one can use the

RECORDTYPE='STREAM'

option to avoid having record descriptor information imbedded in the output file. Examples of C and FORTRAN programs to create station data sets are provided later in this document.

Station Data Descriptor File

The format for the data descriptor file for station data is similar to the format for a gridded data set. An example of a station data descriptor file is:

```
dset  ^ua.reps
dtype  station
stnmap ^ua.map
undef  -999.0
title  Real Time Upper air obs
tdef  10 linear 12z18jan1992 12hr
vars 12
slp 0  99  SLP
ts  0  99  Temps
ds  0  99  Dewpoints
us  0  99  U Winds
vs  0  99  V Winds
z   1  99  Heights
t   1  99  Temps
d   1  99  Dewpoints
u   1  99  U Winds
v   1  99  V Winds
endvars
```

Note the differences between this descriptor file and a grid descriptor file:

DTYPE record -- specify a data type of: station.
STNMAP record -- gives the file name of the station mapping file. This file is created by the stnmap utility, which will be described later.
XDEF, YDEF, ZDEF records -- not specified.
TDEF record -- describes the time grouping interval and the number of time groups in the file.
VAR records -- surface variables must come first, and are given a zero for the number-of-levels field. Level dependent variables are listed after the surface variables, and are given a one in the number-of-levels field.

Byte-ordering control for station data files: You may specify byte ordering (big_endian, or little_endian) for station data files using the OPTIONS record. The stnmap utility, and GrADS, will perform the necessary conversion. Station map files must be created on the machine where they are to be used.

STNMAP Utility

Once the data set has been written, and the descriptor file created, you should then create the station map file by running the stnmap utility. This utility writes out hash table and/or link list information that allows GrADS to access the report data more

efficiently. The utility will prompt for the name of the data descriptor file.

If you change the data file -- perhaps by appending another time group -- you will also have to change the descriptor file to reflect the changes -- the new number of times for example -- and then rerun the stnmap utility.

A.3. Examples of Creating a Gridded Data Set.

On a workstation, the binary GrADS data sets need to be created as a 'stream' data set, ie, it should not have the normal FORTRAN record descriptor words imbedded in it. This can be done from FORTRAN using direct access I/O:

```
      REAL  Z(72,46,16)
        .
        .
      OPEN (8,FILE='grads.dat',FORM='UNFORMATTED',
     &   ACCESS='DIRECT',RECL=72*46)
        .
        .
      IREC=1
      DO 10 I=1,18
       WRITE (8,REC=IREC) ((Z(J,K,I),J=1,72),K=1,46)
       IREC=IREC+1
   10 CONTINUE
```

This example writes out 16 levels of one variable to a file in direct access format. We are really writing the data out sequentially, and using direct access to avoid having the record descriptor words written. There may be options in your compiler to do this more directly, or you may wish to write the data using a C program.

Another simple sample might be:

```
      REAL X(100)
      DO 10 I=1,100
       X(I)=I
   10 CONTINUE
      OPEN (8,FILE='samp.dat',FORM='UNFORMATTED',ACCESS='DIRECT',
     &       RECL=100)
      WRITE (8,REC=1) X
      STOP
      END
```

The associated descriptor file:

```
DSET samp.dat
```

```
TITLE Sample Data Set
UNDEF -9.99E33
XDEF 100 LINEAR 1 1
YDEF 1 LINEAR 1 1
ZDEF 1 LINEAR 1 1
TDEF 1 LINEAR 1JAN2000 1DY
VARS 1
x  0  99  100 Data Points
ENDVARS
```

Once created, you can use this data set to experiment with GrADS data functions, such as:

```
display sin(x/50)
```

A.4. Examples of Creating Station Data Sets.

Lets say you have a data set with monthly rainfall:

```
Year Month Stid      Lat    Lon     Rainfall

1980   1  QQQ       34.3   -85.5   123.3
1980   1  RRR       44.2   -84.5    87.1
1980   1  SSS       22.4   -83.5   412.8
1980   1  TTT       33.4   -82.5    23.3
1980   2  QQQ       34.3   -85.5   145.1
1980   2  RRR       44.2   -84.5   871.4
1980   2  SSS       22.4   -83.5   223.1
1980   2  TTT       33.4   -82.5    45.5
 .
 .
 .
```

A FORTRAN program in DEC FORTRAN to write this data set in GrADS format might be:

```
      CHARACTER*8 STID
C
      OPEN (8,NAME='rain.ch')
      OPEN (10,NAME='rain.dat',FORM='UNFORMATTED',
     &         RECORDTYPE='STREAM')
C
      IFLAG = 0
```

```
C
C  Read and Write
C
10    READ (8,9000,END=90) IYEAR,IMONTH,STID,RLAT,RLON,RVAL
9000  FORMAT (I4,3X,I2,2X,A8,3F8.1)
      IF (IFLAG.EQ.0) THEN
        IFLAG = 1
        IYROLD = IYEAR
        IMNOLD = IMONTH
      ENDIF
C
C  If new time group, write time group terminator.
C  Assuming no empty time groups.
C
      IF (IYROLD.NE.IYEAR.OR.IMNOLD.NE.IMONTH) THEN
        NLEV = 0
        WRITE (10) STID,RLAT,RLON,TIM,NLEV,NFLAG
      ENDIF
      IYROLD = IYEAR
      IMNOLD = IMONTH
C
C  Write this report
C
      TIM = 0.0
      NLEV = 1
      NFLAG = 1
      WRITE (10) STID,RLAT,RLON,TIM,NLEV,NFLAG
      WRITE (10) RVAL
      GO TO 10
C
C  On end of file write last time group terminator.
C
90    CONTINUE
      NLEV = 0
      WRITE (10) STID,RLAT,RLON,TIM,NLEV,NFLAG
      STOP
      END
```

For a different compiler, you would need to determine the appropriate OPEN statement to write a stream data set.

An equivalent C program might be:

```
#include <stdio.h>
```

```
/* Structure that describes a report header in a stn file */

struct rpthdr {
 char id[8];         /* Character station id          */
 float lat;          /* Latitude of report            */
 float lon;          /* Longitude of report           */
 float t;            /* Time in relative grid units   */
 int  nlev;          /* Number of levels following    */
 int flag;           /* Level independent var set flag */
} hdr;

main () {
FILE *ifile, *ofile;
char rec[80];
int flag,year,month,yrsav,mnsav,i;
float val;

 /* Open files */

 ifile = fopen ("rain.ch","r");
 ofile = fopen ("rain.dat","wb");
 if (ifile==NULL || ofile==NULL) {
  printf ("Error opening files\n");
  return;
 }

 /* Read, write loop */

 flag = 1;
 while (fgets(rec,79,ifile)!=NULL) {

  /* Format conversion */

  sscanf (rec,"%i %i ",&year,&month);
  sscanf (rec+20," %g %g %g",&hdr.lat,&hdr.lon,&val);
  for (i=0; i<8; i++) hdr.id[i] = rec[i+11];

  /* Time group terminator if needed */

  if (flag) {
   yrsav = year;
   mnsav = month;
   flag = 0;
  }
  if (yrsav!=year || mnsav!=month) {
   hdr.nlev = 0;
```

```
      fwrite (&hdr,sizeof(struct rpthdr), 1, ofile);
    }
    yrsav = year;
    mnsav = month;

    /* Write this report */

    hdr.nlev = 1;
    hdr.flag = 1;
    hdr.t = 0.0;
    fwrite (&hdr,sizeof(struct rpthdr), 1, ofile);
    fwrite (&val,sizeof(float), 1, ofile);
  }
  hdr.nlev = 0;
  fwrite (&hdr,sizeof(struct rpthdr), 1, ofile);
}
```

Once the binary data file has been written, create the descriptor file. It would look something like this:

```
dset   rain.dat
dtype  station
stnmap rain.map
undef  -999.0
title  Rainfall
tdef  12 linear jan1980 1mo
vars 1
p  0  99  Rainfall
endvars
```

Then run the stnmap utility to create the station map file. You can then open and display this data from within GrADS.

A.5. Examples of 'stations.ua' and 'stations.sfc' Data Sets.

The upper air station file is an ASCII list that includes the Stid, Lat, Lon, and station title as shown here:

```
NZY  32.7 -117.2 North Island
MIR  32.9 -117.1 San Diego
...   ...   ...   ...
...   ...   ...   ...
...   ...   ...   ...
EDW  34.9 -117.9 Edwards AFB
LAS  36.6 -116.0 Las Vegas
OAK  37.7 -122.2 Oakland
```

The surface station file is also an ASCII list, but only includes the Stid, Lat, and Lon as shown here:

```
ACV  41.0 -124.1
BAB  39.1 -121.4
 ...   ...    ...
 ...   ...    ...
UKI  39.1 -123.2
WMC  40.9 -117.8
YUM  32.7 -114.6
```

This Page Intentionally Left Blank.

Appendix B: NEBS Scripts

This Page Intentionally Left Blank.

B.1. NEBS.GS

```
* NRL Environmental data Browsing System (NEBS)
*
* This browser works only with GrADS version 1.4.1
* Created by Gary G. Love at NRL Monterey, Dec 93.
* Based on SJLBROWSE.GS developed by Steve J. Lord at NMC.
*  Time clock was obtained from Mike Fiorino at NMC.
*
* Must use with output.gs to print or make video frames.
* Must customize output.gs to use own devices, paths, and programs.
*
* The input files can be specified on the GrADS command line:
*
*    run nebs.gs gridfile surfacefile upperairfile
*
*   If surfacefile is specified or input to query,
*   file station.sfc must be present and specify station
*   3-character name, latitude, and longitude:
*         LAX  34.0 -118.4
*
*   If upperairfile is specified or input to query,
*   file station.ua must be present and specify station
*   3-character name, latitude, longitude, and title:
*         MIR  32.9 -117.1 San Diego
*
*   Surfacefile and upperairfile should be specified on the
*   command line only if both are present.
*
function gb(parm)
'reinit'

*  Check for command line input and request if missing
_ftype="Grid"
parm1=subwrd(parm,1)
if (parm1="")
  '!ls *.ctl'
  say "Enter control file name for GRID data."
  pull _fname.1
else
  _fname.1=parm1
endif

* Open gridded file
'open '_fname.1
res=result
say res
'q file 1'
_stnnum=0
_sfcnum=0

* Check if surface and/or upper air data available and desired
parm2=subwrd(parm,2)
parm3=subwrd(parm,3)
if (parm3="")
 if (parm2="")
  say "Enter control file name for SURFACE data,"
  say "IF available and desired. ELSE {return}"
  pull _fname.2
  say "Enter control file name for UPPER AIR data."
  say "IF available and desired. ELSE {return}"
  if (_fname.2="")
   pull _fname.2
   if (_fname.2!="")
    _ftype="Station"
   endif
  else
   pull _fname.3
   if (_fname.3="")
     _ftype="Surface"
   else
     _ftype="Both"
   endif
  endif
 else
  _fname.2=parm2
  _ftype="Surface"
  say "Enter control file name for UPPER AIR data."
  say "IF available and desired. ELSE {return}"
  pull _fname.3
  if (_fname.3!="")
   _ftype="Both"
  endif
 endif
else
  _fname.2=parm2
  _fname.3=parm3
  _ftype="Both"
endif

* Open surface and/or upper air files if requested
if (_ftype="Surface")
 'open '_fname.2
 dum=sublin(result,2)
 res=subwrd(dum,2)
 if(res!="Error:")
  _fn=2
  'q file 2'
  rc=readsfc()
  rc=fnddt(_fname.2)
  _dtime.2=subwrd(rc,1)
```

```
 endif
endif
if (_ftype="Station")
 'open '_fname.2
 dum=sublin(result,2)
 res=subwrd(dum,2)
 if(res!="Error:")
  _fn=2
  'q file 2'
  rc=readua()
  rc=fnddt(_fname.2)
  _dtime.2=subwrd(rc,1)
 endif
endif
if (_ftype="Both")
 'open '_fname.3
 dum=sublin(result,2)
 res=subwrd(dum,2)
 if(res!="Error:")
  _fn=2
  'q file 2'
  rc=readua()
  rc=fnddt(_fname.3)
  _dtime.2=subwrd(rc,1)
 endif
 'open '_fname.2
 dum=sublin(result,2)
 res=subwrd(dum,2)
 if(res!="Error:")
  _fn=3
  'q file 3'
  rc=readsfc()
  rc=fnddt(_fname.2)
  _dtime.3=subwrd(rc,1)
 endif
endif

* Initial setup of parameters
rc=resetd()
rc=butncons()
rc=colors()
rc=clear()
rc=dobutn()
prfile="browse.prn."_name
say "print file is "prfile
rc=dodisp()
```

```
*** MAIN LOOP ***
* Warning: it is absolutely necessary that the following
*   functions be called in exactly this order:
while (1)
 if(_pick!=2)
  rc = qbutton ()
  rc = qbar()
 endif
 if (_btn=99); break; endif
 if (_btn=97); _out=1; 'enable print 'prfile; endif
 if (_btn=96); rc=resetd(); endif
 _display=0
 if (_mtyp>0); 'clear'; else; rc = clear(); endif
 rc = monitor()
 rc = display()
 if (_mtyp>0); 'disable print'; rc=output(); _mtyp=0; endif
 rc = dobutn()
 _supp=0
endwhile
*******************

if(_nprint = 0)
 say "No maps have been printed."
endif
say "BROWSE SESSION ENDED."
'quit'


*** Functions ***

function display()
* Regulates the display of grid, profile and surface plots
* Warning: it is absolutely necessary that the following
*   functions be called in exactly this order:
 if (_mtyp>0); case=_case; endif
 if (_pick=1&_styp=3); case=1; endif
 if (_pick=2&_styp=3); case=2; endif
 if (_pick=0&_styp=3&_disurf=1&_btnN="2")
  if(_btn<0&_xpos<_timbxll&_xpos>_dimbxur); case=3; endif
 endif
 if (_tflag=0&(_btn=98|(_btn<0 & _btnN="3"))); case=4; endif
 if (case=1)
  rc=dosurf()
  _supp=1
  _case=1
 endif
 if (case=2)
  _pick=0
  rc=disurf()
  _display=0
```

```
 _supp=1
 _case=2
endif
if (case=3)
 rc=disurf()
 _display=0
 _supp=1
 _case=3
endif
if (case=4)
 rc = dodisp()
 _supp=1
 _case=4
endif
_tflag=0
return

function dobutn()
* Performs button actions
* Warning: it is absolutely necessary that the following
*   functions be called in exactly this order:
rc=vbutton()
rc=miscbutn()
rc=sbutton()
rc=stbutton()
rc=gxbutton()
rc=pbutton()
rc=tbar()
rc=dbar()
return

function resetd()
* Resets display and variables
'reset'
_clev=1
_clk=0
_cvar=1
_dclick=2
_dens=5
_display=0
_disurf=0
_disgrp=0
_entr=0
_gxtp=1
k=1
while (k<=7)
 _gxuse.k=0
 k=k+1
endwhile
_mtyp=0
_nchvr=8
_nfram=0
_nprint=0
_out=0
k=1
while (k<=_sfcnum)
 _picked.k=0
 k=k+1
endwhile
_pick=0
_sfp=0
_shad=0
_skip=1
_styp=3
k=1
while (k<=20)
 _vtyp.k=0
 _vgxt.k=0
 _cint.k=""
 _cmax.k=""
 _cmin.k=""
 _lbl.k=1
 _mrkr.k=0
 _styl.k=1
 k=k+1
endwhile
_station=0
_supp=1
_surface=0
_tclick = 2
_tflag=0
_trans=0
_t=1
_t1=1
_t2=1
_uvar=0
_vvar=0
_x1=1
_x2=_xnum
_y1=1
_y2=_ynum
_z1=1
_z2=1
'set dfile 1'
rc=getbasic()
return

function getbasic()
* Get info on the data file
'q file 1'
```

```
say result
res = result
dum = sublin(res,1)
i = 4; _maptitle = ''
while (i < 10)
  _maptitle = _maptitle' 'subwrd(dum,i)
  i = i+1
endwhile
dum = sublin(res,5)
_znum = subwrd(dum,9)
_tnum = subwrd(dum,12)
_xnum = subwrd(dum,3)
_ynum = subwrd(dum,6)
dum = sublin(res,6)
_vnum = subwrd(dum,5)
'set x 1 '_xnum
'set y 1 '_ynum
'q dims'
say result
dum = sublin(result,2)
_lnlo = subwrd(dum,6)
_lnhi = subwrd(dum,8)
dum = sublin(result,3)
_ltlo = subwrd(dum,6)
_lthi = subwrd(dum,8)
* get time limits in character form
'set t '1
_tc1 = mydate()
'set t '_tnum
_tc2 = mydate()
* get time interval
rc=fnddt(_fname.1)
_dtime.1=subwrd(rc,1)
_tinc=_dtime.1
_tdcl=_tinc
_nint=_dtime.1/_tinc
_tnums=(_tnum-1)*_nint+1
* get world values for each lat and lon
i = 1
_lons = ''
while (i<=_xnum)
 'set x 'i
 _lons = _lons % ' ' % subwrd(result,4)
 i = i + 1
endwhile
_x2=_xnum
i = 1
_lats = ''
while (i<=_ynum)
 'set y 'i
 _lats = _lats % ' ' % subwrd(result,4)
 i = i + 1
endwhile
_y2=_ynum
* get world values for each level
i = 1
_levs = ''
while (i<=_znum)
 'set z 'i
 _levs = _levs % ' ' % subwrd(result,4)
 i = i + 1
endwhile
* set labeling of lats, lons, levs, and time
i = 1
_xmark=""
while (i<=_xnum)
 rng=int(_lnhi-_lnlo)
 lab = subwrd(_lons,i)
 rc=ckinc(lab,rng)
 if (rc=0)
  _xmark=_xmark" "1
 else
  _xmark=_xmark" "0
 endif
 i = i + 1
endwhile
i = 1
_ymark=""
while (i<=_ynum)
 rng=int(_lthi-_ltlo)
 lab = subwrd(_lats,i)
 rc=ckinc(lab,rng)
 if (rc=0)
  _ymark=_ymark" "1
 else
  _ymark=_ymark" "0
 endif
 i = i + 1
endwhile
i = 1
_zmark=""
while (i<=_znum)
 rng=_znum
 lab = subwrd(_levs,i)
 rc=ckinc(lab,rng)
 if (rc=0)
  _zmark=_zmark" "1
 else
  _zmark=_zmark" "0
 endif
```

```
 i = i + 1
 endwhile
 i = 1
 _tmark=""
 while (i<=(_tnum-1)*_nint+1)
  rc=mod(i-1,_nint)
  if (rc=0)
   _tmark=_tmark" "1
  else
   if (_tdel<10&_iunit="MN")
    rc=mod(i-1,10)
    if (rc=0)
     _tmark=_tmark" "0
    else
     _tmark=_tmark" "s
    endif
   else
    _tmark=_tmark" "0
   endif
  endif
  i = i + 1
 endwhile
* get each variable abbreviation and level designation
 i = 0
 _vars = ''
 while (i<_vnum)
  dum = sublin(rcs,i+7)
  var=subwrd(dum,1)
  _vars = _vars % ' ' % var
  j = i + 1
  if(var="pmsl"|var="slp")
   _vtyp.j=1
   _vgxt.j=1
  endif
  _nlevs.j = subwrd(dum,2)
  dums=4
  if(substr(var,1,1)="u")
   dums=5
   _uvar=j
  endif
  if(substr(var,1,1)="v")
   dums=5
   _vvar=j
  endif
  iw=dums
  _vname.j=""
  while (subwrd(dum,iw) !="")
   _vname.j = _vname.j%" "%subwrd(dum,iw)
   iw=iw+1
  endwhile
   i = i + 1
  endwhile
* if winds available, define divergence and curl variables here
* and create for selected dimensions in function disp().
  if(_uvar!=0&_vvar!=0)
   _vars=_vars % " hdiv vort"
   _vnum=_vnum+1
   _vname._vnum="horizontal divergence"
   _hd = _vnum
   _vnum=_vnum+1
   _vname._vnum="relative vorticity"
   _vt = _vnum
  endif
* set variable contour limits, line styles and markers
  i = 0
  while (i<_vnum)
   i = i + 1
   rc=style(i)
   rc=marker(i)
   var=subwrd(_vars,i)
   rc=getemp(var,_cint.i,_cmax.i,_cmin.i)
   _cint.i=_ci
   _cmax.i=_cmx
   _cmin.i=_cmn
  endwhile
 return

 function butncons()
 * Set button parameters
  'q gxinfo'
  dum=sublin(result,2)
  _xpur=subwrd(dum,4)
  _ypur=subwrd(dum,6)
  _xpll=0.0
  _ypll=0.0
 * size of Menu Area
  _mwide=1.33
  _mhigh=1.5
 * height of and width of Graphic Output Boxes (inches)
  _gxbw=_mwide/1
  _gxbh=_mhigh/5
 * width of time bar display and height of time labels (inches)
  _timbwide=0.50
  _timlhigh=0.2
 * width of xyz bar display and height of labels (inches)
  _bwide=0.60
  _lhigh=0.2
 * dimensions of quit, display, reset, and print boxes
  _qbw=1.22
  _qbh=0.5
```

```
_dbw=1.33
_dbh=0.5
_pbw=1.22
_pbh=0.5
_rbw=1.22
_rbh=0.5
* dimensions of plot, image, and movie buttons
_mbw=1.22
_mbh=0.25
* height and width of section buttons
_sbw=1.22
_sbh=0.4
* height and width of station buttons
_stbw=0.8
_stbh=0.3
* height and width of variables boxes
_vbw=0.55
_vbh=0.4
_topbyll=_ypur-_vbh
if(_ftype="Station"|_ftype="Both")
 _vbh = 0.3
 _topbyll=_ypur-_vbh-_stbh
endif
* border for all buttons
_bord=1/80
return

function miscbutn()
* Draw misc contol buttons
'set button 0 90 91 92 6'
if (_out = 0)
 i = 99
 qbx=_qbw/2+_bord*2
 qby=_qbh/2+0.08
 'draw button 'i' 'qbx' 'qby' '_qbw' '_qbh' QUIT
 i = 97
 pbx=_qbw/2+(_qbw-_pbw)/2+_bord*2
 pby=_qbh/2+0.1+_dbh+_bord
 'draw button 'i' 'pbx' 'pby' '_pbw' '_pbh' PRINT
endif
i = 98
dbx=_xpur-_dbw/2-4*_bord
dby=_dbh/2+0.1
'draw button 'i' 'dbx' 'dby' '_dbw' '_dbh' DISPLAY'
i = 96
rbx=_rbw/2+2*_bord
rby=_ypur-_rbh/2-2*_bord
'draw button 'i' 'rbx' 'rby' '_rbw' '_rbh' RESET'
return
```

```
function gxbutton()
* Draw graphic control buttons
_gxtype="contour shaded vector barb stream"
_gxtlab="contour shaded vector barbs stream"
delx=_gxbw-_bord
dely=_gxbh-_bord*2
xbut=_xpur-_gxbw/2-_bord*4
ybut=_ypur-_gxbh/2-_bord
nd=5; ng=1
while ng <= nd
str=subwrd(_gxtlab,ng)
if(_gxtp != ng)
 'set button 0 90 91 92 6'
 'draw button 6'ng' 'xbut' 'ybut' 'delx' 'dely' 'str
else
 'set button 1 90 92 91 6'
 'draw button 6'ng' 'xbut' 'ybut' 'delx' 'dely' 'str
endif
ybut=ybut-_gxbh
ng=ng+1
endwhile
return

function pbutton()
* Draw print control buttons
if (_out = 1)
 _prt="BW.PLOT C.PLOT MOVIE VIDEO"
 delx=_mbw-_bord
 dely=_mbh-_bord
 xbut=_mbw/2+_bord*3
 ybut=_ypll+3.5*_mbh+_bord+0.1
 i = 1; j = 1
 while (j = 1)
  while (i <= 4)
   str=subwrd(_prt,i)
   if (i != _mtyp)
    'set button 0 90 91 92 6'
    'draw button 'i+55' 'xbut' 'ybut' 'delx' 'dely' 'str
   else
    'set button 1 90 92 91 6'
    'draw button 'i+55' 'xbut' 'ybut' 'delx' 'dely' 'str
   endif
   ybut=ybut-_mbh-_bord
   i = i + 1
  endwhile
  j = 0
 endwhile
endif
_out=0
return
```

```
function qbutton()
* Query buttons and define actions
** program waits here for next instruction **
 'q pos'
 _xpos = subwrd(result,3)
 _ypos = subwrd(result,4)
 _btn = subwrd(result,7)
 _event = subwrd(result,6)
 _btnN = subwrd(result,5)
 if (_event=0)
  _btn = -999
 endif
 n=_btn
 e=0.25
 if(_display=1 & _styp=3 & _btnN="1")
  if(_xpos<=_parxmx+e&_xpos>=_parxmn-e&_ypos<=_parymx+e&_ypos>=_parymn-1.0)
   if(_xpos<=_parxmx-e&_xpos>=_parxmn+e&_ypos<=_parymx-e&_ypos>=_parymn+e)
    rc=zoomin()
   else
    rc=zoomout()
   endif
   _btn=98
  endif
 endif
 if(_xpos<=_parxmx&_xpos>=_parxmn&_ypos<=_parymx&_ypos>=_parymn&_display=1)
  if(_ftype="Station"); return; endif
  if(_surface=1)
   if(_btnN="2"); _pick=1; endif
  endif
  if(_ftype="Grid")
   if(_btnN="2"); return; endif
  endif
 else
  if(_xpos<_timbxll & _xpos>_dimbxur & _surface=1 & _display=0)
   if (_btn<0 & _btnN="2"); _disurf=1; endif
  endif
* toggle variable button
  if (n>=1 & n<=20 & _btnN!="3")
   _entr=1
   if (_vtyp.n != 1)
    _vtyp.n=1
   else
    _vtyp.n=0
   endif
  endif
* exclusively set make output button
  if (n=56); _mtyp=1; endif;
  if (n=57); _mtyp=2; endif;
  if (n=58); _mtyp=3; endif;
  if (n=59); _mtyp=4; endif;
* exclusively set graph type button
  if (n>=61 & n<=65)
   _entr=2
  endif
* set contour or fill button
  if (n=61); _gxtp=1; endif;
  if (n=62); _gxtp=2; endif;
* set vect, barb or stream button
  if (n=63); _gxtp=3; endif;
  if (n=64); _gxtp=4; endif;
  if (n=65); _gxtp=5; endif;
* toggle surface
  if (n=70)
   if(_surface=0)
    _surface=1
   else
    _surface=0
   endif
  endif
* exclusively set station button
  if (n>=71 & n<=89)
   if(n!=_station+70)
    _station=n-70
   else
    _station=0
   endif
  endif
* n=90 is RETURN button
* exclusively set section button
  if (n=91); _styp=1; endif;
  if (n=92); _styp=2; endif;
  if (n=93); _styp=3; endif;
* n=95 is ENTER button
  if (n=95); _entr=4; endif;
* n=96 is RESET button
* n=97 is PRINT button
  if (n=97); _out=1; endif;
* n=98 is DISPLAY button
* if (_btn<0 & _btnN=3); n=98; endif
* n=99 is QUIT button
 endif
return

function sbutton()
* Draw spatial dimension buttons
 _sec.1="Lon"
 _sec.2="Lat"
 _sec.3="Lev"
 delx=_sbw-_bord
 dely=_sbh-_bord
```

```
xbut=_sbw/2+2*_bord
ybut=_ypur-_rbh-_sbh/2-_bord-0.05
i = 1
while (i <= 3)
 str=_sec.i
 if (i != _styp)
  'set button 0 90 91 92 6'
  'draw button 9'i' 'xbut' 'ybut' 'delx' 'dely' 'str
 else
  'set button 1 90 92 91 6'
  'draw button 9'i' 'xbut' 'ybut' 'delx' 'dely' 'str
 endif
 ybut=ybut-_sbh
 i = i + 1
endwhile
return

function stbutton()
* Draw station selection buttons
if(_ftype!="Grid")
 xmin=_rbw+4*_bord
 xmax=_xpur-_gxbw-4*_bord
 if(_ftype="Station")
  if (_stbw*_stnnum>(xmax-xmin))
   stbw=(xmax-xmin)/_stnnum
  else
   stbw=_stbw
  endif
  delx=stbw-_bord
  dely=_stbh-_bord
  xbut=(xmax+xmin)/2-stbw*_stnnum/2+stbw/2
  ybut=_ypur-_vbh-_stbh/2-_bord
  i = 1
  while (i<=_stnnum)
   str=_stid.i
   if (i!=_station)
    'set button 0 90 91 92 6'
    'draw button '70+i' 'xbut' 'ybut' 'delx' 'dely' 'str
   else
    'set button 1 90 92 91 6'
    'draw button '70+i' 'xbut' 'ybut' 'delx' 'dely' 'str
   endif
   xbut=xbut+stbw
   i = i + 1
  endwhile
 endif
 if(_ftype="Surface")
  delx=2*_stbw-_bord
  dely=_stbh-_bord
  xbut=(xmax+xmin)/2
  ybut=_ypur-_vbh-_stbh/2-_bord
  str="SURFACE"
  if (_surface=0)
   'set button 0 90 91 92 6'
   'draw button 70 'xbut' 'ybut' 'delx' 'dely' 'str
  else
   'set button 1 90 92 91 6'
   'draw button 70 'xbut' 'ybut' 'delx' 'dely' 'str
  endif
 endif
 if(_ftype="Both")
  if (_stbw*(_stnnum+2)>(xmax-xmin))
   stbw=(xmax-xmin)/(_stnnum+2)
  else
   stbw=_stbw
  endif
  delx=stbw-_bord
  dely=_stbh-_bord
  xbut=(xmax+xmin)/2-stbw*_stnnum/2-_bord
  ybut=_ypur-_vbh-_stbh/2-_bord
  str="SURFACE"
  if (_surface=0)
   'set button 0 90 91 92 6'
   'draw button 70 'xbut' 'ybut' '2*delx' 'dely' 'str
  else
   'set button 1 90 92 91 6'
   'draw button 70 'xbut' 'ybut' '2*delx' 'dely' 'str
  endif
  xbut=xbut+stbw*3/2
  i = 1
  while (i<=_stnnum)
   str=_stid.i
   if (i!=_station)
    'set button 0 90 91 92 6'
    'draw button '70+i' 'xbut' 'ybut' 'delx' 'dely' 'str
   else
    'set button 1 90 92 91 6'
    'draw button '70+i' 'xbut' 'ybut' 'delx' 'dely' 'str
   endif
   xbut=xbut+stbw
   i = i + 1
  endwhile
 endif
endif
return

function vbutton()
* Draw variable selection buttons
 xmin=_rbw+4*_bord
 xmax=_xpur-_gxbw-4*_bord
```

```
if (_vbw*_vnum>(xmax-xmin))
 vbw=(xmax-xmin)/(_vnum)
else
 vbw=_vbw
endif
delx=vbw-_bord
dely=_vbh-_bord
xbut=(xmax+xmin)/2-vbw*_vnum/2+vbw/2-_bord
ybut=_ypur-_vbh/2-_bord
i = 1
while (i<=_vnum)
 str=subwrd(_vars,i)
 if(_btn=i & _btnN="3")
  rc=vmenu(i,xbut,ybut,delx+0.3,dely,str)
 endif
 xbut=xbut+vbw
 i = i + 1
endwhile
xbut=(xmax+xmin)/2-vbw*_vnum/2+vbw/2
i = 1
while (i<=_vnum)
 str=subwrd(_vars,i)
 if (_vtyp.i != 1)
  'set button 0 90 91 92 6'
  'draw button 'i' 'xbut' 'ybut' 'delx' 'dely' 'str
 else
  'set button 1 90 92 91 6'
  'draw button 'i' 'xbut' 'ybut' 'delx' 'dely' 'str
 endif
 xbut=xbut+vbw
 i = i + 1
endwhile
return

function tmenu()
* Draw time customization menu
rc= clear()
y = 0.5*(_timbyll+_timbyur)
x = 0.5*(_timbxll+_timbxur)-1.5
c = 0.35
'set strsiz 0.2'
'set string 1 r 8'
'draw string 'x' 'y+2.0*c' Time clock'
'draw string 'x' 'y' Time interval'
tinc=_tdel % ' ' % _iunit
'draw string 'x' 'y-c' is 'tinc
'set button 0 90 91 92 6'
* get new time interval
j=1
while (j=1)
 'draw button 91 'x+0.5' 'y' 'c' 'c' >'
 'draw button 92 'x+0.5' 'y-c' 'c' 'c' <'
 if(_clk!=1)
  'draw button 93 'x+0.5' 'y+2.0*c' 'c' 'c' Off'
 endif
 if(_clk!=0)
  'set button 1 90 92 91 6'
  'draw button 93 'x+0.5' 'y+2.0*c' 'c' 'c' On'
  'set button 0 90 91 92 6'
 endif
 'set string 95 r 8'
 'q pos'
 btn = subwrd(result,7)
 btnN = subwrd(result,5)
 if(btnN="3")
  if(btn=91)
   'draw string 'x' 'y-c'  is '_tdel % ' ' % _iunit
   rc = timinc(1)
   if(_tinc<_dtime.1); clr=7; else; clr=1; endif
   'set string 'clr' r 8'
   'draw string 'x' 'y-c'  is '_tdel % ' ' % _iunit
  endif
  if(btn=92)
   'draw string 'x' 'y-c'  is '_tdel % ' ' % _iunit
   rc = timinc(-1)
   if(_tinc<_dtime.1); clr=7; else; clr=1; endif
   'set string 'clr' r 8'
   'draw string 'x' 'y-c'  is '_tdel % ' ' % _iunit
  endif
  if(btn=93)
   if(_clk!=0); _clk=0; else; _clk=1; endif
  endif
 else
  j=0
  rc = clear()
 endif
 if(_tinc<_dtime.1)
  'set string 7 r 8'
  'draw string 'x' 'y+c' Interpolated '
 else
  'set string 95 r 8'
  'draw string 'x' 'y+c' Interpolated '
 endif
endwhile
* define new _t1 and _t2
nint=_dtime.1/_tinc
_t1 = (_t1-1)/_nint+1
_t2 = (_t2-1)/_nint+1
_t1 = int((_t1-1)*nint+1)
_t2 = int((_t2-1)*nint+1)
```

```
_nint=nint
_tnums=(_tnum-1)*_nint+1
* create tick mark vector
i = 1
_tmark=""
while (i<=(_tnum-1)*_nint+1)
 rc=mod(i-1,_nint)
 if (rc=0)
  _tmark=_tmark" "1
 else
  if (_tdel<10&_iunit="MN")
   rc=mod(i-1,10)
   if (rc=0)
    _tmark=_tmark" "0
   else
    _tmark=_tmark" "s
   endif
  else
   _tmark=_tmark" "0
  endif
 endif
 i = i + 1
endwhile
return

function vmenu(i,xbut,ybut,delx,dely,str)
* Draw variable customization menu
_supp=1
a=0.5; b=0.3; c=0.3; d=0.75
x=xbut
y=ybut-0.5
'draw button 'i' 'x' 'y+0.5' 'delx' 'dely' 'str
'set button 0 90 91 92 6'
'set strsiz 0.2'
'set string 1 c 8'
'draw string 'x' 'y-c' CONTOUR'
'draw string 'x' 'y-5*d-c' LINE'
if((substr(str,1,1)="u"|substr(str,1,1)="v")&str!="vort")
 'draw string 'x' 'y-7*d-c' VECTOR'
endif
rc=style()
rc=marker()
rc=getemp(str,_cint.i,_cmax.i,_cmin.i)
'set strsiz 0.15'
'set string 1 r 6'
'draw string 'x+a' 'y-1*d-c' Labels'
'draw string 'x+a' 'y-2*d' 'str' interval'
'draw string 'x+b' 'y-2*d-c' is '_ci
'draw string 'x+a' 'y-3*d' 'str' maximum'
'draw string 'x+b' 'y-3*d-c' is '_cmx
'draw string 'x+a' 'y-4*d' 'str' minimum'
'draw string 'x+b' 'y-4*d-c' is '_cmn
'draw string 'x+a' 'y-6*d' Style '_sty.i
'draw string 'x+a' 'y-6*d-c' Marker '_mkr.i
if((substr(str,1,1)="u"|substr(str,1,1)="v")&str!="vort")
 'draw string 'x+a' 'y-8*d' 'str' grid skip'
 'draw string 'x+b' 'y-8*d-c' is '_skip-1
 'draw string 'x+a' 'y-9*d' 'str' streamline'
 'draw string 'x+a' 'y-9*d-c' density is '_dens
endif
j=1
while (j=1)
 if(_lbl.i!=1)
  'draw button 80 'x+1.0' 'y-1*d-c' 'c' 'c' Off'
 endif
 if(_lbl.i!=0)
  'set button 1 90 92 91 6'
  'draw button 80 'x+1.0' 'y-1*d-c' 'c' 'c' On'
  'set button 0 90 91 92 6'
 endif
 'draw button 81 'x+1.0' 'y-2*d' 'c' 'c' >'
 'draw button 82 'x+1.0' 'y-2*d-c' 'c' 'c' <'
 'draw button 83 'x+1.0' 'y-3*d' 'c' 'c' >'
 'draw button 84 'x+1.0' 'y-3*d-c' 'c' 'c' <'
 'draw button 85 'x+1.0' 'y-4*d' 'c' 'c' >'
 'draw button 86 'x+1.0' 'y-4*d-c' 'c' 'c' <'
 'draw button 87 'x+1.0' 'y-6*d' 'c' 'c' ?'
 'draw button 88 'x+1.0' 'y-6*d-c' 'c' 'c' ?'
 if((substr(str,1,1)="u"|substr(str,1,1)="v")&str!="vort")
  'draw button 89 'x+1.0' 'y-8*d' 'c' 'c' >'
  'draw button 90 'x+1.0' 'y-8*d-c' 'c' 'c' <'
  'draw button 91 'x+1.0' 'y-9*d' 'c' 'c' >'
  'draw button 92 'x+1.0' 'y-9*d-c' 'c' 'c' <'
 endif
 'set string 95 r 6'
 'q pos'
 btn = subwrd(result,7)
 btnN = subwrd(result,5)
 if(btnN="3")
  if(btn=80)
   if(_lbl.i!=0); _lbl.i=0; else; _lbl.i=1; endif
  endif
  if(btn=81)
   'draw string 'x+b' 'y-2*d-c' is '_cint.i
   _cint.i=_cint.i+_cinc
   if(_cint.i>_cmax.i-_cmin.i); _cint.i=_cmax.i-_cmin.i; endif
   'set string 7 r 6'
   'draw string 'x+b' 'y-2*d-c' is '_cint.i
  endif
  if(btn=82)
```

```
 'draw string 'x+b' 'y-2*d-c'  is '_cint.i
 _cint.i=_cint.i-_cinc
 if(_cint.i<=0); _cint.i=_cinc; endif
 if((_cmax.i-_cmin.i)/_cint.i>99); _cint.i=_cint.i+_cinc; endif
 'set string 7 r 6'
 'draw string 'x+b' 'y-2*d-c'  is '_cint.i
endif
if(btn=83)
 'draw string 'x+b' 'y-3*d-c'  is '_cmax.i
 _cmax.i=_cmax.i+5*_cinc
 if((_cmax.i-_cmin.i)/_cint.i>99); _cmax.i=_cmax.i-5*_cinc; endif
 'set string 7 r 6'
 'draw string 'x+b' 'y-3*d-c'  is '_cmax.i
endif
if(btn=84)
 'draw string 'x+b' 'y-3*d-c'  is '_cmax.i
 _cmax.i=_cmax.i-5*_cinc
 if(_cmax.i<=_cmin.i); _cmax.i=_cmin.i+5*_cinc; endif
 'set string 7 r 6'
 'draw string 'x+b' 'y-3*d-c'  is '_cmax.i
endif
if(btn=85)
 'draw string 'x+b' 'y-4*d-c'  is '_cmin.i
 _cmin.i=_cmin.i+5*_cinc
 if(_cmax.i<=_cmin.i); _cmin.i=_cmax.i-5*_cinc; endif
 'set string 7 r 6'
 'draw string 'x+b' 'y-4*d-c'  is '_cmin.i
endif
if(btn=86)
 'draw string 'x+b' 'y-4*d-c'  is '_cmin.i
 _cmin.i=_cmin.i-5*_cinc
 if((_cmax.i-_cmin.i)/_cint.i>99); _cmin.i=_cmin.i+5*_cinc; endif
 'set string 7 r 6'
'draw string 'x+b' 'y-4*d-c'  is '_cmin.i
endif
if(btn=87)
 'draw string 'x+a' 'y-6*d' Style '_sty.i
 _styl.i=_styl.i+1
 if(_styl.i=5); _styl.i=1; endif
 rc=style(i)
 'set string 7 r 6'
 'draw string 'x+a' 'y-6*d' Style '_sty.i
endif
if(btn=88)
 'draw string 'x+a' 'y-6*d-c' Marker '_mkr.i
 _mrkr.i=_mrkr.i+1
 if(_mrkr.i=6); _mrkr.i=0; endif
 rc=marker(i)
 'set string 7 r 6'
 'draw string 'x+a' 'y-6*d-c' Marker '_mkr.i
endif
if(btn=89)
 'draw string 'x+b' 'y-8*d-c'  is '_skip-1
 _skip=_skip+1
 'set string 7 r 6'
 'draw string 'x+b' 'y-8*d-c'  is '_skip-1
endif
if(btn=90)
 'draw string 'x+b' 'y-8*d-c'  is '_skip-1
 _skip=_skip-1
 if(_skip<=1); _skip=1; endif
 'set string 7 r 6'
 'draw string 'x+b' 'y-8*d-c'  is '_skip-1
endif
if(btn=91)
 'draw string 'x+a' 'y-9*d-c' density is '_dens
 _dens=_dens+1
 if(_dens>=10); _dens=10; endif
 'set string 7 r 6'
 'draw string 'x+a' 'y-9*d-c' density is '_dens
endif
if(btn=92)
 'draw string 'x+a' 'y-9*d-c' density is '_dens
 _dens=_dens-1
 if(_dens<=1); _dens=1; endif
 'set string 7 r 6'
 'draw string 'x+a' 'y-9*d-c' density is '_dens
endif
if(i=_uvar)
 _cint._vvar=_cint._uvar
 _cmax._vvar=_cmax._uvar
 _cmin._vvar=_cmin._uvar
endif
if(i=_vvar)
 _cint._uvar=_cint._vvar
 _cmax._uvar=_cmax._vvar
 _cmin._uvar=_cmin._vvar
endif
else
 j=0
 rc= clear()
endif
endwhile
if(_cmax._uvar<_cint._uvar)
 cmax=_cint._uvar
else
 cmax=_cmax._uvar
endif
return
```

```
function setparea()
* Set plot area for each plot type
 bordlft=0.8
 bordbtm=1.5
 bordrit=0.5
 bordtop=0.6
 if(_plottyp=0)
  bordlft=1.0
  bordbtm=1.2
  bordrit=0.5
  bordtop=0.9
 endif
 if(_plottyp=2)
  bordlft=1.0
  bordbtm=1.2
  bordrit=0.75
  bordtop=0.9
 endif
 if(_plottyp=1|_plottyp=3)
  bordlft=2.0
  bordbtm=1.2
  bordrit=0.5
  bordtop=0.9
 endif
 if(_plottyp=4|_plottyp=5)
  bordlft=1.2
  bordbtm=1.5
  bordrit=0.5
  bordtop=0.9
 endif
 if(_plottyp=6)
  bordlft=2.0
  bordbtm=1.5
  bordrit=0.5
  bordtop=0.9
 endif
 if(_station!=0)
  bordlft=1.2
  bordbtm=1.0
  bordrit=0.6
  bordtop=0.9
 endif
 _parxmx=_timbxll-bordrit
 _parymx=_topbyll-bordtop
 _parxmn=_dimbxur+bordlft
 _parymn=_xpll+bordbtm
 'set parea '_parxmn' '_parxmx' '_parymn' '_parymx
 'set line 0'
 'draw recf '_parxmn' '_parymn' '_parxmx' '_parymx
return
```

```
function bar(x, y, dx, dy, n, p1, p2, ticks)
* Selection bar draw widget
 dx2=dx/2
 dx6=dx/6
 dx12=dx/12
 yd = dy/(n-1)
 ylo = y-dy/2
 yhi = y+dy/2
 'set line 0 1 4'
 'draw line 'x' 'ylo' 'x' 'yhi
 i = 1
 while (i<=n)
  y = ylo + (i-1)*yd
  tk = subwrd(ticks,i)
  if (i>=p1 & i<=p2)
   'set line 92 1 5.5'
  else
   'set line 0 1 4'
  endif
  if (tk="1")
   'draw line '%(x-dx6)%' 'y' '%(x+dx6)%' 'y
  else
   if (_tinc!=_dtime.1&x>_timbxll)
    'set line 41 1 4'
   endif
   if (tk="0"); 'draw line '%(x-dx12)%' 'y' '%(x+dx12)%' 'y; endif
  endif
  i = i + 1
 endwhile
 y1=ylo+(p1-1)*yd
 y2=ylo+(p2-1)*yd
 'set line 92 1 12'
 'draw line 'x' 'y1' 'x' 'y2
return

function dbar()
* Draw dimension bar
* set selected dimension
 if (_styp=1)
  _nam=_lons
  num=_xnum
  lab1 = "Lon"
  mark=_xmark
  p1=_x1
  p2=_x2
 endif
 if (_styp=2)
  _nam=_lats
  num=_ynum
  lab1 = "Lat"
```

```
 mark=_ymark
 p1=_y1
 p2=_y2
endif
if (_styp=3)
 _nam=_levs
 num=_znum
 lab1 = "Lev"
 mark=_zmark
 p1=_z1
 p2=_z2
endif
_dlev = p1
* size dimension bar
nrow=2
nchr=3
lh2=0.5*_lhigh
xhi=_xpll+_bwide
xlo=_xpll
ylo = _qbh+_pbh+0.3+nrow*_lhigh
yhi = _ypur-_rbh-_sbh*3-0.25-nrow*_lhigh
delx=0.35
dely=0.01
_dimbxll=xlo+delx
_dimbxur=xhi+delx
_dimbyll=ylo-dely
_dimbyur=yhi-dely
if(num = 1); return; endif
* draw dimension bar
x = 0.5*(xhi+xlo)+delx
dx = _bwide
y = 0.5*(ylo+yhi)
dy = yhi-ylo
xll=_dimbxll-0.3
yll=_dimbyll-0.55
xur=_dimbxur+0.3
yur=_dimbyur+0.55
'set line 90'
'draw recf 'xll' 'yll' 'xur' 'yur
'set line 92 1 6'
'draw line 'xll' 'yll' 'xll' 'yur
'draw line 'xll' 'yur' 'xur' 'yur
'set line 91 1 6'
'draw line 'xll' 'yll' 'xur' 'yll
'draw line 'xur' 'yll' 'xur' 'yur
rc = bar(x, y, dx, dy, num, p1, p2, mark)
* label dimension bar
str=lh2
if(nchr*str > dx)
 str=dx/nchr
endif
'set string 1 c 8'
'set strsiz 'str
'draw string 'x' '%(ylo-2.0*_lhigh+lh2)%' 'lab1
'draw string 'x' '%(yhi+2.0*_lhigh-lh2)%' 'lab1
* label selected dimension values
str=0.8*str
yd = dy/(num-1)
i = 1
while (i<=num)
 'set strsiz 'str
 ylab = ylo + (i-1)*yd
 lab2 = subwrd(_nam,i)
 ilab2 = lab2
 mkit = subwrd(mark,i)
 if(mkit="1")
  ilab2=int(lab2)
  'set string 1 l 3'
  xlab = x+dx/4
  'draw string 'xlab' 'ylab' 'ilab2
 endif
 if(i=p1|i=p2)
  if(lab2-int(lab2)!=0); ilab2=int(lab2*100)/100; endif
  'set string 1 r 3'
  xlab = x-dx/4
  'draw string 'xlab' 'ylab' 'ilab2
 endif
* display current settings
 if(i=_styp)
  'set string 7 l 6'
 else
  'set string 15 l 6'
 endif
 if(_xpur<_ypur)
  'set strsiz 0.12'
  xx=_xpur-7.0
  yy=_ypur-6.0
 else
  'set strsiz 0.15'
  xx=_xpur-9.0
  yy=_ypur-4.5
 endif
 if(_supp!=1)
  if(i=1)
   'draw string 'xx' 'yy+2.0' Lon = 'subwrd(_lons,_x1)
   if(_x1!=_x2)
    'draw string 'xx+0.5' 'yy+1.6' to 'subwrd(_lons,_x2)
   endif
  endif
  if(i=2)
```

```
  'draw string 'xx' 'yy' Lat = 'subwrd(_lats,_y1)
  if(_y1!=_y2)
   'draw string 'xx+0.5' 'yy-0.4' to 'subwrd(_lats,_y2)
  endif
 endif
 if(i=3)
  'draw string 'xx' 'yy-2.0' Lev = 'subwrd(_levs,_z1)
  if(_z1!=_z2)
   'draw string 'xx+0.5' 'yy-2.4' to 'subwrd(_levs,_z2)
  endif
 endif
endif
i = i + 1
endwhile
return

function qbar()
* Query selection bar
bord = 0.1
*  "dimension bar has been chosen"
if (_xpos<=_dimbxur & _xpos>=_dimbyll-bord & _ypos<=_dimbyur+bord)
* identify selected dimension
 if (_styp=1)
  num=_xnum
  p1=_x1
  p2=_x2
 endif
 if (_styp=2)
  num=_ynum
  p1=_y1
  p2=_y2
 endif
 if (_styp=3)
  num=_znum
  p1=_z1
  p2=_z2
 endif
* get selected dimension index
 td = (_dimbyur-_dimbyll)/(num-1)
 dim = (_ypos-_dimbyll)/td
 dim = int(dim+1.5)
 if (dim<1); dim = 1; endif
 if (dim>num); dim = num; endif
* set selected dimension
 if (_dclick=1 & _btnN="2")
  _dclick = 2
  if (p1<dim)
   p2 = dim
  else
   p1 = dim
  endif
 else
  _dclick = 1
  p1 = dim
  p2 = dim
 endif
 if (_styp=1)
  _x1=p1
  _x2=p2
 endif
 if (_styp=2)
  _y1=p1
  _y2=p2
 endif
 if (_styp=3)
  _z1=p1
  _z2=p2
 endif
endif
* "time bar has been chosen"
if (_xpos>=_timbxll & _xpos>=_timbyll-bord & _ypos<=_timbyur+bord)
* get tmenu to modify _tinc and _nint
 y = 0.5*(_timbyll+_timbyur)
 x = 0.5*(_timbxll+_timbxur)
 x = x -1.5*_timbwidc/8-_timlhigh/4
 if (x>=_xpos-0.1&x<=_xpos+0.3&y>=_ypos-0.5&y<=_ypos+0.5&_btnN="3")
  rc=tmenu()
  rc=tdeflbl(0)
  _tflag = 1
 else
* get selected times
  _tnums=(_tnum-1)*_nint+1
  td = (_timbyur-_timbyll)/(_tnums-1)
  tim = (_ypos-_timbyll)/td
  tim = int(tim+1.5)
  if (tim<1); tim = 1; endif
  if (tim>_tnums); tim = _tnums; endif
* set _t1 and _t2
  if (_tclick=1 & _btnN="2")
   _tclick = 2
   if (_t1<tim)
    _t2 = tim
   else
    _t1 = tim
   endif
  else
   _tclick = 1
   _t1 = tim
   _t2 = tim
  endif
```

```
 endif
 endif
return

function tbar()
* Draw time bar
* size time bar
 nrowtim=3
 nchtim=8
 timlh2=0.5*_timlhigh
 xhi=_xpur
 xlo=_xpur-_timbwide
 ylo = _dbh+0.2+nrowtim*_timlhigh
 yhi = _ypur-_mhigh-0.2-nrowtim*_timlhigh
 delx=0.45
 dely=0.05
 _timbxll=xlo-delx
 _timbxur=xhi-delx
 _timbyll=ylo+dely
 _timbyur=yhi+dely
 if(_tnum = 1); return; endif;
* draw time bar
 x = 0.5*(xhi+xlo)-delx
 dx = _timbwide
 y = 0.5*(ylo+yhi)
 dy = yhi-ylo
 xll=_timbxll-0.4
 yll=_timbyll-0.7
 xur=_timbxur+0.4
 yur=_timbyur+0.65
 'set line 90'
 'draw recf 'xll' 'yll' 'xur' 'yur
 'set line 92 1 6'
 'draw line 'xll' 'yll' 'xll' 'yur
 'draw line 'xll' 'yur' 'xur' 'yur
 'set line 91 1 6'
 'draw line 'xll' 'yll' 'xur' 'yll
 'draw line 'xur' 'yll' 'xur' 'yur
 rc = bar(x, y, dx, dy, _tnums, _t1, _t2, _tmark)
* label first and last times
 timstr=timlh2
 if(nchtim*timstr > 2*dx)
  timstr=2*dx/nchtim
 endif
 'set string 1 c 8'
 'set strsiz 'timstr
 p1 = subwrd(_tc1,1)
 p2 = subwrd(_tc1,2) % ' ' % subwrd(_tc1,3)
 p3 = subwrd(_tc1,4)
 'draw string 'x' '%(ylo-_timlhigh+timlh2)%' 'p1
 'draw string 'x' '%(ylo-2.0*_timlhigh+timlh2)%' 'p2
 'draw string 'x' '%(ylo-3.0*_timlhigh+timlh2)%' 'p3
 p4 = subwrd(_tc2,1)
 p5 = subwrd(_tc2,2) % ' ' % subwrd(_tc2,3)
 p6 = subwrd(_tc2,4)
 'draw string 'x' '%(yhi+_timlhigh-timlh2)%' 'p4
 'draw string 'x' '%(yhi+2.0*_timlhigh-timlh2)%' 'p5
 'draw string 'x' '%(yhi+3.0*_timlhigh-timlh2)%' 'p6
* label selected times
 yd = dy/(_tnums-1)
 i = 1
 'set strsiz 'timstr*0.8
 while (i<=_tnums)
  if(i=_t1|i=_t2)
   rc=int((i-1)/_nint+1.0)
   'set t 'rc
   tc = mydate()
   p1 = tlabel(i)
   p2 = subwrd(tc,2) % ' ' % subwrd(tc,3)
   ylab = ylo + (i-1)*yd
   xlab=x-dx/3
   'set string 1 r 3'
   'draw string 'xlab' 'ylab' 'p1
   xlab=x+dx/3
   'set string 1 l 3'
   'draw string 'xlab' 'ylab' 'p2
   if(i=_t1); t1=p1' 'p2; endif
   if(i=_t2); t2=p1' 'p2; endif
  endif
  i = i + 1
 endwhile
* display current settings
 if(i=_styp)
  'set string 7 l 6'
 else
  'set string 15 l 6'
 endif
 if(_xpur<_ypur)
  'set strsiz 0.12'
  xx=_xpur-4.0
  yy=_ypur-6.0
 else
  'set strsiz 0.15'
  xx=_xpur-5.0
  yy=_ypur-4.5
 endif
 if(_supp!=1)
  'draw string 'xx' 'yy' Time = 't1
  if(_t1!=_t2)
   'draw string 'xx+0.5' 'yy-0.4' to 't2
```

```
 endif
 endif
* add tdef label
 if (_tinc=_dtime.1)
  rc=tdeflbl(1)
 else
  rc=tdeflbl(7)
 endif
return

function tdeflbl(clr)
*  Temporary fix for displaying time interval (needs an upgrade)
 y = 0.5*(_timbyll+_timbyur)
 x = 0.5*(_timbxll+_timbxur)
 x = x -1.5*_timbwide/8-_timlhigh/4
 'set strsiz 0.12'
 'set string 'clr' c 3 270'
 tinc=_tdel % ' ' % _iunit
 if(_tinc<_dtime.1); tinc='Interpolated '%tinc; endif
 'draw string 'x' 'y' 'tinc
 'set string 'clr' c 3 0'
return

function tlabel(val)
* Control units for time labels
 ht = getime(val)
 hr = int(ht)
 hf = ht - hr
 mn = int(60*hf+0.05)
 mf = 60*hf - mn
 sc = int(60*mf+0.45)
 if (hr<10); hr="0"%hr; endif
 if (mn<10); mn="0"%mn; endif
 if (sc<10); sc="0"%sc; endif
 if(_tunit="MN"|_tunit="mn")
  p = hr%':'%mn%':'%sc%'Z'
 endif
 if(_tunit="HR"|_tunit="hr")
  p = hr%':'%mn%'Z'
 endif
 if(_tunit="DY"|_tunit="dy")
  hr = int(24*ht+0.05)
  if (hr<10); hr="0"%hr; endif
  p = hr%'Z'
 endif
return p

function fnddt(fname)
* Temporary fix to determine the delta time between data sets
 while (1)
  res=read(fname)
  dum=sublin(res,2)
  if(subwrd(dum,1)="tdef"|subwrd(dum,1)="TDEF"); break; endif
 endwhile
 rc=close (fname)
 dt=subwrd(dum,5)
 i=1
 while(substr(dt,i,1)>='0' & substr(dt,i,1)<='9')
  i=i+1
 endwhile
 num=substr(dt,1,i-1)
 _tunit=substr(dt,i,99)
 _iunit=_tunit
 str=num%" "_tunit
return str

function mydate ()
* Reformat the GrADS date/time into something more readable
 'query time'
 sres = subwrd(result,3)
 i = 1
 while (substr(sres,i,1)!='Z')
  i = i + 1
 endwhile
 hour = substr(sres,1,i)
 isav = i
 i = i + 1
 while (substr(sres,i,1)>='0' & substr(sres,i,1)<='9')
  i = i + 1
 endwhile
 day = substr(sres,isav+1,i-isav-1)
 month = substr(sres,i,3)
 year = substr(sres,i+3,4)
return (hour' 'day' 'month' 'year)

function getime(val)
* Determine time as set
 'query time'
 dtg=subwrd(result,3)
 if(substr(dtg,3,1)=":"); min=substr(dtg,4,2); endif
 if(substr(dtg,3,1)="Z"); min=0; endif
 if(_tunit="DY"|_tunit="dy")
  tp1=substr(dtg,1,2)
  tp2=tp1+_dtime.1
 endif
 if(_tunit="HR"|_tunit="hr")
  tp1=substr(dtg,1,2)
  tp2=tp1+_dtime.1
 endif
 if(_tunit="MN"|_tunit="mn")
```

```
  tp1=substr(dtg,1,2)+min/60
  tp2=tp1+_dtime.1/60
 endif
 if (tp2=0); tp2=24; endif
 b=mod((val-1),_nint)/_nint
 a=1.0-b
 p=a*tp1+b*tp2
return p

function ckinc(val,n)
* Set a reasonable axis label increment
*---------------
* this is a cluge to handle irregular grid data
 dlon=(_lnhi-_lnlo)/(_xnum+1)
 dlat=(_lthi-_ltlo)/(_ynum+1)
 if (dlon>dlat)
  e = dlat
 else
  e = dlon
 endif
*---------------
 ival = int(val)
 if (val-ival>elival-val>e); ival=val; endif;
 if (n<=10); return(0); endif;
 if (n>10 & n<=20)
  rc=mod(ival,2)
  return(rc)
 endif
 if (n>20 & n<=50)
  rc=mod(ival,5)
  return(rc)
 endif
 if (n>50 & n<=120)
  rc=mod(ival,10)
  return(rc)
 endif
 if (n>120 & n<=360)
  rc=mod(ival,30)
  return(rc)
 endif
 if (n>360)
  rc=mod(ival,50)
  return(rc)
 endif

function mod(i0,inc)
* Calculate the modulus
 if(inc!=0)
  imod=int(i0/inc)
 else
  imod=int(i0/1)
 endif
 imod=i0-imod*inc
return(imod)

function int(i0)
* Calculate integer
 i=0
 while(i<12)
  i=i+1
  if(substr(i0,i,1)='.')
   i0=substr(i0,1,i-1)
   break
  endif
 endwhile
return(i0)

function timinc(val)
* Set time increment for selected units
 denom=int(1/_tinc+0.5)
 tinc=_tinc
 if(_tunit="MN"|_tunit="mn"); mod=60; endif
 if(_tunit="HR"|_tunit="hr"); mod=60; endif
 if(_tunit="DY"|_tunit="dy"); mod=24; endif
 if(_tunit="MO"|_tunit="mo"); mod=30; endif
 if(_tunit="YR"|_tunit="yr"); mod=12; endif
 i=1
 while (i=1)
  if (val>0)
   if (tinc<1)
    denom=denom-1
    if(denom<1); denom=1; endif
    if(mod(mod,denom)=0); _tinc=1/denom; break; endif
   else
    tinc = tinc +1
    if (tinc>=_dtime.1); _tinc=_dtime.1; break; endif
    if (mod(_dtime.1,tinc)=0); _tinc=tinc; break; endif
   endif
  endif
  if (val<0)
   if (tinc<=1)
    denom=denom+1
    if(denom>mod); denom=mod; endif
    if(mod(mod,denom)=0); _tinc=1/denom; break; endif
   else
    tinc = tinc -1
    if(mod(_dtime.1,tinc)=0); _tinc=tinc; break; endif
   endif
  endif
 endwhile
```

```
if(_tinc<1&_tunit="MN"); _tdel=int(_tinc*60+0.5); _iunit="SC"; endif
if(_tinc>=1&_tunit="MN"); _tdel=_tinc; _iunit="MN"; endif
if(_tinc<1&_tunit="HR"); _tdel=int(_tinc*60+0.5); _iunit="MN"; endif
if(_tinc>=1&_tunit="HR"); _tdel=_tinc; _iunit="HR"; endif
if(_tinc<1&_tunit="DY"); _tdel=int(_tinc*24+0.5); _iunit="HR"; endif
if(_tinc>=1&_tunit="DY"); _tdel=_tinc; _iunit="DY"; endif
if(_tinc<1&_tunit="MO"); _tdel=int(_tinc*30+0.5); _iunit="DY"; endif
if(_tinc>=1&_tunit="MO"); _tdel=_tinc; _iunit="MO"; endif
if(_tinc<1&_tunit="YR"); _tdel=int(_tinc*12+0.5); _iunit="MO"; endif
if(_tinc>=1&_tunit="YR"); _tdel=_tinc; _iunit="YR"; endif
if(_tinc<1&_tunit="mn"); _tdel=int(_tinc*60+0.5); _iunit="sc"; endif
if(_tinc>=1&_tunit="mn"); _tdel=_tinc; _iunit="mn"; endif
if(_tinc<1&_tunit="hr"); _tdel=int(_tinc*60+0.5); _iunit="mn"; endif
if(_tinc>=1&_tunit="hr"); _tdel=_tinc; _iunit="hr"; endif
if(_tinc<1&_tunit="dy"); _tdel=int(_tinc*24+0.5); _iunit="hr"; endif
if(_tinc>=1&_tunit="dy"); _tdel=_tinc; _iunit="dy"; endif
if(_tinc<1&_tunit="mo"); _tdel=int(_tinc*30+0.5); _iunit="dy"; endif
if(_tinc>=1&_tunit="mo"); _tdel=_tinc; _iunit="mo"; endif
if(_tinc<1&_tunit="yr"); _tdel=int(_tinc*12+0.5); _iunit="mo"; endif
if(_tinc>=1&_tunit="yr"); _tdel=_tinc; _iunit="yr"; endif
return

function monitor()
* Monitor multiple usage of shaded selection
if(_entr=1 & _vtyp._btn=1)
 if(_gxuse._gxtp=0 | _gxtp=1 | _btn=_uvar | _btn=_vvar)
  _vgxt._btn=_gxtp
  _gxuse._gxtp=_btn
  if(_gxtp=2); _shad=_btn; endif
 else
  _vtyp._btn=0
  'set string 1 c 6'
  'set strsiz 0.2'
  str1="""%subwrd(_gxtlab,_gxtp)"""
  str2=subwrd(_vars,_gxuse._gxtp)
  str=str1%" has been used for "%str2
  'draw string '_xpur/2' '_ypur-2*(_vbh+_stbh)' 'str
  str="SELECT A NEW GRAPH TYPE"
  'draw string '_xpur/2' '_ypur-4*(_vbh+_stbh)' 'str
  _supp=1
 endif
endif
if(_entr=1 & _vtyp._btn=0)
 if(_vgxt._btn=2)
  _shad=0
  _gxuse.2=0
 endif
 _vgxt._btn=0
 _gxuse._gxtp=0
endif
 _entr=0
return

function dodisp()
* Control type of plot
 'set x '_x1' '_x2
 'set y '_y1' '_y2
 'set z '_z1' '_z2
 if (_styp=1); _dv="x"; endif
 if (_styp=2); _dv="y"; endif
 if (_styp=3); _dv="z"; endif
 if (_styp=3& _x1<_x2&_y1<_y2)
  'q dims'
  dum = sublin(result,2)
  lnlo = subwrd(dum,6)
  lnhi = subwrd(dum,8)
  dum = sublin(result,3)
  ltlo = subwrd(dum,6)
  lthi = subwrd(dum,8)
  'set mpdset lowres'
  if(lnhi-lnlo<90&lthi-ltlo<45); 'set mpdset mres'; endif
  if(lnhi-lnlo<20&lthi-ltlo<10); 'set mpdset hires'; endif
 endif
 t1=int((_t1-1)/_nint+1.0)
 t2=int((_t2-1)/_nint+1.0)
 if(_t1<_t2&_t2<_tnums&mod((_t2-1)*_tinc,_dtime.1)!=0); t2=t2+1; endif
 _t = _t1;  _tt1=t1; _tt2=t2
 'set t '_tt1' '_tt2
 rc=plottype()
 rc=setparea()
 if (_plottyp=7 & _station=0)
  'set string 1 c 6'
  'set strsiz 0.18'
  str = 'Select single level of Lat, Lon, Lev, or Time'
  'draw string '_xpur/2' '_ypur/2-0.5' 'str
  str = 'Click Mouse Button to Continue'
  'draw string '_xpur/2' '_ypur/2-1.0' 'str
  'q pos'
  rc = clear()
  return
 endif
 if (_station!=0)
  if (_z1=_z2)
   'set z 1 '_znum
  endif
  'set lat '_slat._station
  'set lon '_slon._station
  if (_t1<_t2)
   rc = dotime()
  else
```

```
  rc = disprof()
  _nfram=1
 endif
else
 if(_plottyp=0|_plottyp=3|_plottyp=4|_plottyp=6)
  rc = disp()
  _nfram=1
 endif
 if (_plottyp=1|_plottyp=2)
  if (_t1=_t2)
   rc = dodims()
  else
   rc = dotime()
  endif
 endif
 if(_plottyp=5)
  rc = dodims()
 endif
endif
_tclick = 2
_dclick = 2
return

function dotime()
* Animate in time
 if (_t1<_t2)
  'set dbuff on'
  _t = _t1
  while (_t<=_t2)
   'set t 'int((_t-1)/_nint+1.0)
   if(_station!=0)
    rc = disprof()
   else
    rc = disp()
   endif
   _t = _t + 1
   if (_t>_t2)
    'set string 1 c 6'
    'set strsiz 0.2'
    str = 'Click Mouse Button to Continue'
    'draw string '_xpur/2' '_ypur-0.25' 'str
    _display=0
   endif
   'swap'
  endwhile
  if (_mtyp=0); 'q pos'; rc=clear(); endif
  'set dbuff off'
  _nfram=_t2-_t1+1
 else
  _nfram=1
 endif
 _t = _t1
 'set t '_tt1
return

function dodims()
* Animate in x, y, or z
 if (_styp=1)
  p1=_x1
  p2=_x2
 endif
 if (_styp=2)
  p1=_y1
  p2=_y2
 endif
 if (_styp=3)
  p1=_z1
  p2=_z2
 endif
 if (p1<p2)
  'set dbuff on'
  p = p1
  while (p<=p2)
   'set '_dv' 'p
   _dlev = p
   rc = disp()
   p = p + 1
   if (p>p2)
    'set string 1 c 6'
    'set strsiz 0.2'
    str = 'Click Mouse Button to Continue'
    'draw string '_xpur/2' '_ypur-0.25' 'str
    _display=0
   endif
   'swap'
  endwhile
  if (_mtyp=0); 'q pos'; rc=clear(); endif
  'set dbuff off'
  _nfram=p2-p1+1
 else
  rc = disp()
  _nfram=1
 endif
 'set '_dv' 'p1
return

function dosurf()
* Display and select surface stations
* define interpolation values
 n=_fn
```

```
b=mod((_t-1),_nint)/_nint
a=1.0-b
to=int(b*_dtime.1/_dtime.n)
d=b*_dtime.1/_dtime.n-to
c=1.0-d
'set grads off'
'set ccolor 0'
'set cthick 6'
'set gxout model'
'set ccolor 30'
'set cthick 6'
'set gxout model'
if (_t=_tnums)
 'd u.'n'(t+'to');v.'n'(t+'to')'
else
 arg1 = c'*u.'n'(t+'to')+'d'*u.'n'(t+'to+1')'
 arg2 = c'*v.'n'(t+'to')+'d'*v.'n'(t+'to+1')'
 'd 'arg1';'arg2
endif
'set line 31'
p=0
while (p<=_sfcnum)
 p=p+1
 if(_picked.p=1)
  'q w2xy '_sfclon.p' '_sfclat.p
  sfx=subwrd(result,3)
  sfy=subwrd(result,6)
  'draw mark 3 'sfx' 'sfy' 0.07'
  'set string 31 bl 2'
  'set strsiz 0.08'
  'draw string 'sfx+0.08' 'sfy+0.08' '_sfcnam.p
 endif
endwhile
while (_pick=1)
 inplot=0
 if(_xpos<=_parxmx&_xpos>=_parxmn&_ypos<=_parymx&_ypos>=_parymn); inplot=1;
endif
 if(_btnN="1"); _pick=0; rc=clear(); break; endif
 if(_btnN="2"&inplot=1); rc=picksfc(_xpos,_ypos); endif
 if(_btnN="2"&inplot=0); _pick=2; break; endif
 if(_btnN="3"); _pick=0; break; endif
*  'program waits here for pick instruction'
 'q pos'
 _xpos = subwrd(result,3)
 _ypos = subwrd(result,4)
 _btnN = subwrd(result,5)
endwhile
return
```

```
function disp()
* Function to display a map
'set grads off'
'set csmooth on'
'set poli on'
'set map 1 1 6'
'set dignum 0'
'set digsize .08'
* set up to plot shaded variable first
if(_vtyp._uvar=1 & _vtyp._vvar=1 & _shad=_vvar); _shad=_uvar; endif
if(_shad!=0)
 didshad=0
else
 didshad=1
endif
* create any defined variables here so they can be interpolated *
if(_uvar!=0&_vvar!=0&(_vtyp._hd=1|_vtyp._vt=1)&_plottyp=2)
 'set t 'int((_t-1)/_nint+1.0)' 'int((_t+1)/_nint+1.0)
 uvar=subwrd(_vars,_uvar)
 vvar=subwrd(_vars,_vvar)
 'define hdiv = hdivg('uvar','vvar')'
 'define vort = hcurl('uvar','vvar')'
 'set t 'int((_t-1)/_nint+1.0)
endif
* prescan variables
i=1
cn=0
while (i<=_vnum)
 gxtp=_vgxt.i
 if(gxtp=1); cn=cn+1; endif
 i=i+1
endwhile
* variable plot loop
a=1; b=0
i=1; j=0; k=1; l=0
doit=0
while (i<=_vnum)
 if(i=_shad & didshad=0); doit=1; endif
 if(i!=_shad & didshad=1 & _vtyp.i=1); doit=1; endif
 if(i=_vvar & _vgxt._vvar=2 & didshad=1); doit=0; endif
 if(doit=1)
  j=j+1
  gxtp=_vgxt.i
  if(gxtp=1&didshad=1); l=l+1; endif
  zz=subwrd(_gxtype,gxtp)
  'set gxout 'zz
* set plot exceptions
  if((_plottyp=4|_plottyp=5)&_z1=_z2)
   'set xyrev on'
  endif
```

```
  if(_plottyp=5&_styp=3)
   'set xyrev on'
  endif
  if(_plottyp=1|_plottyp=2)
   'set '_dv' '_dlev
  endif
* set line and symbol type
  if(_plottyp=1|_plottyp=3|_plottyp=6)
   if(_styl.i=4); styl=5; else; styl=_styl.i; endif
   'set cstyle 'styl
   'set cmark '_mrkr.i
  endif
* set labels and y-axis colors
  levlab=subwrd(_nam,_dlev)
  'set grid off'
  if(_plottyp=2|_plottyp>=4)
   'set ylopts 1'
   if(didshad=1&_shad!=0&gxtp<=2)
    k=subwrd(_colors,l)
    'set ccolor 'k
   endif
   if(cn>1&j>1&gxtp<=2)
    k=subwrd(_colors,j-1)
    'set ccolor 'k
   endif
  endif
  if(_plottyp=0|_plottyp=1|_plottyp=3|_plottyp=6)
   k=subwrd(_colors,j)
   'set ccolor 'k
   'set ylopts 'k
  endif
* define plot variable and settings
  var=subwrd(_vars,i)
  levels=getemp(var,_cint.i,_cmax.i,_cmin.i)
  if(_lbl.i=0)
   'set clab off'
  else
   'set clab on'
  endif
  'set cthick 6'
* display variable
  if (_plottyp=1|_plottyp=2)
*   define time interpolation values
   b=mod((_t-1),_nint)/_nint
   a=1.0-b
   if (i=_uvar & _vtyp._uvar=1 & _vtyp._vvar=1)
     vvar=vecvar(var,gxtp,_skip,a,b,t1,t2)
    'd 'vvar
    i = i + 1
   else
    if(b=0)
     'd 'var
    else
     'd 'a'*'var'+'b'*'var'(t+1)'
    endif
   endif
  else
   if (i=_uvar & _vtyp._uvar=1 & _vtyp._vvar=1)
    b=0
    vvar=vecvar(var,gxtp,_skip,a,b,t1,t2)
    'd 'vvar
    i = i + 1
   else
    'd 'var
   endif
  endif
* overlay wind isotach contours on wind vector plot
  if(i=_vvar&_vgxt._vvar!=_vgxt._uvar&_vtyp._uvar=1&_vtyp._vvar=1&didshad=1)
   vgxt=subwrd(_gxtype,_vgxt._vvar)
   gxtp=_vgxt._vvar
   'set gxout 'vgxt
   var=subwrd(_vars,i)
   'set clevs 'levels
   if ((_plottyp=1|_plottyp=2)&b=0)
    post=vecvar(var,gxtp,_skip,a,b,t1,t2)
    'd 'post
   else
    post=vecvar(var,gxtp,_skip,a,b,t1,t2)
    'd 'post
   endif
  endif
* overlay black contour boundary
  if(_shad!=0 & didshad=0 & (_plottyp=2|_plottyp=4|_plottyp=5))
   'set gxout contour'
   'set clevs 'levels
   'set ccolor 0'
   if(i=_vvar & _vtyp._uvar=1 & _vtyp._vvar=1)
    'd 'vvar
   else
    if (b=0)
     'd 'var
    else
     'd 'a'*'var'+'b'*'var'(t+1)'
    endif
   endif
  endif
* overlay surface stations
  if(_surface=1 & _styp=3 & _plottyp=2)
   rc=dosurf()
   _disurf=1
```

```
  endif
* labels in plots
  res=result
  'set string 'k' c 6'
  'set strsiz 0.15'
  xval=_parxmx - 1.5
  yval=_parymx - 0.5
  if((_plottyp=2|_plottyp=4|_plottyp=5)&_shad!=0)
   'set string 0 c 6'
  endif
  if(_plottyp=1|_plottyp=2|_plottyp=5)
   if (_x1=_x2 & _styp!=1)
    if(_shad=0); 'set string 1 c 6'; endif
    'draw string 'xval' 'yval' lon = 'subwrd(_lons,_x1)
   endif
   if (_y1=_y2 & _styp!=2)
    if(_shad=0); 'set string 1 c 6'; endif
    'draw string 'xval' 'yval' lat = 'subwrd(_lats,_y1)
   endif
   if (_z1=_z2 & _styp!=3)
    if (_nlevs.i = 1|_nlevs.i = 0)
     'draw string 'xval' 'yval' lev = surface'
    else
     'draw string 'xval' '%yval-0.3*j%' lev = 'subwrd(_levs,_z1)
    endif
   endif
  else
   if(_plottyp!=0&_plottyp!=5)
    if(_styp=1)
     if(_plottyp!=4)
      if (_nlevs.i = 1|_nlevs.i = 0)
       'draw string 'xval' '%yval-0.3*j%' lev = surface'
      else
       'draw string 'xval' '%yval-0.3*j%' lev = 'subwrd(_levs,_z1)
      endif
     endif
     if(_shad=0|_plottyp=1|_plottyp=3|_plottyp=6)
      'set string 1 c 6'
     endif
     'draw string 'xval' 'yval' lat = 'subwrd(_lats,_y1)
    endif
    if(_styp=2)
     if(_plottyp!=4)
      if (_nlevs.i = 1|_nlevs.i = 0)
       'draw string 'xval' '%yval-0.3*j%' lev = surface'
      else
       'draw string 'xval' '%yval-0.3*j%' lev = 'subwrd(_levs,_z1)
      endif
     endif
     if(_shad=0|_plottyp=1|_plottyp=3|_plottyp=6)
      'set string 1 c 6'
     endif
     'draw string 'xval' 'yval' lon = 'subwrd(_lons,_x1)
    endif
    if(_styp=3)
     if(_shad=0|_plottyp=1|_plottyp=3|_plottyp=6)
      'set string 1 c 6'
     endif
     'draw string 'xval' 'yval' lon = 'subwrd(_lons,_x1)
     'draw string 'xval' '%yval-0.3%' lat = 'subwrd(_lats,_y1)
    endif
   endif
  endif
* print single values (_plottyp=0)
  xval = (_parxmn+_parxmx)/2
  yval = _parymx - 0.5
  if (_plottyp=0)
   'set string 1 c 6'
   'set strsiz 0.15'
   'draw string 'xval' 'yval' lon = 'subwrd(_lons,_x1)
   'draw string 'xval' 'yval-0.3' lat = 'subwrd(_lats,_y1)
   'draw string 'xval' 'yval-0.6' lev = 'subwrd(_levs,_z1)
   yval = 5.5-j*0.2
   'set string 'k' c 6'
   'draw string 'xval' 'yval' 'subwrd(res,4)
  endif
* title at top of map
  if(_plottyp!=0)
   'draw title '_maptitle
  endif
* labels at bottom of plot
  'q gxinfo'
  dum = sublin(result,3)
  xlo = subwrd(dum,4)
  xhi = subwrd(dum,6)
  dum = sublin(result,4)
  ylo = subwrd(dum,4)
  yhi = subwrd(dum,6)
  rc=mydate()
  p1=tlabel(_t)
  verdat=p1 %' '%subwrd(rc,2)%' '%subwrd(rc,3)%' '%subwrd(rc,4)
  varnam=_vname.i
  if (_styp=1)
   levlab = "lon "%levlab
  endif
  if (_styp=2)
   levlab = "lat "%levlab
  endif
  if (_styp=3)
   levlab = "lev "%levlab
```

```
 endif
 if ((_nlevs.i = 1|_nlevs.i = 0)&_styp=3)
  levlab="surface"
 endif
 if(_plottyp=3)
  if (_styp=1)
   levlab = "lon"
  endif
  if (_styp=2)
   levlab = "lat"
  endif
  if (_styp=3)
   levlab = "lev"
  endif
 endif
 if(_plottyp=4)
  if (_styp!=3)
   if (_nlevs.i = 1|_nlevs.i = 0)
    levlab = "lev surface"
   else
    levlab = "lev "%subwrd(_levs,_z1)
   endif
  endif
  if (_styp=3)
   if (_nlevs.i = 1|_nlevs.i = 0)
    levlab = "lev surface"
   endif
  endif
 endif
 alllab=varnam%"  "%levlab%"  "%verdat
 if(_plottyp=3)
  levlab=""
 endif
 len=howlong(alllab)
 labsiz=(xhi-xlo)/len
 if(labsiz > 0.12)
  labsiz=0.12
 endif
 lend=howlong(varnam)
 xleft=xlo+lend*labsiz
 rbeg=howlong(verdat)
 xright=xhi-rbeg*labsiz
 xcen=0.5*(xlo+xhi)
 if(xcen < lend)
  xcen=0.5*(xleft+xright)
 endif
 if(_plottyp=0)
  spc=0.8
 else
  spc=-0.1
 endif
 if(_plottyp=2&gxtp!=1); k=1; endif
 if(_plottyp>=4)
  spc=-0.4
 endif
 'set strsiz 'labsiz' 0.12'
 'set string 'k' tr 6'
 'draw string 'xhi' '%(ylo-0.18*j+spc)' 'verdat
 'set string 'k' tl 6'
 'draw string 'xlo' '%(ylo-0.18*j+spc)' 'varnam
 'set string 'k' tc 6'
 'draw string 'xcen' '%(ylo-0.18*j+spc)' 'levlab
 if(_shad!=0 & didshad=0)
  didshad=1
  i=0
 endif
 endif
 i=i+1
 doit=0
endwhile
_display=1
* add tclock if selected
if (_clk!=0); rc=tclock(); endif
* print if selected
if(_mtyp>0); 'print'; endif
return

function disprof()
*  Function to display model and raob profiles
'set grads off'
'set strsiz 0.12'
'set ylopts 1'
* define interpolation values
b=mod((_t-1),_nint)/_nint
a=1.0-b
if(_t=_tnums); a=0; b=1; endif
* plot grid values
i=1
j=0
while (i<=_vnum)
 if(_vtyp.i=1)
  j=j+1
  'set grid off'
  'set cthick 6'
  if(_styl.i=4); styl=5; else; styl=_styl.i; endif
  'set cstyle 'styl
  'set cmark '_mrkr.i
  kc=subwrd(_colors,j)
  'set ccolor 'kc
  var=subwrd(_vars,i)
```

```
 rc=getemp(var,_cint.i,_cmax.i,_cmin.i)
 if(_t<_tnums)
  'd 'a'*'var'+'b'*'var'(t+1)'
 else
  'd 'var
 endif
 'set string 1 l 6'
 x=_parxmn
 y=_parymn-0.6
 'draw string 'x' 'y' Grid data'
 'set string 'kc' l 6'
 'draw string 'x+0.8*j+1.2' 'y' 'var
 endif
 i=i+1
endwhile
* invert height and plot station data
lev2=subwrd(_levs,_z2)
lev1=subwrd(_levs,_z1)
'set lev 'lev2' 'lev1
'set yflip on'
if(_t=_tnums); b=0; endif
to=int(b*_dtime.1/_dtime.2)
i=1
k=0
while (i<=_vnum)
 if(_vtyp.i=1&mod(b*_dtime.1,_dtime.2)=0)
  k=k+1
  j=j+1
  'set cthick 6'
  if(_styl.i=4); styl=5; else; styl=_styl.i; endif
  'set cstyle 'styl
  'set cmark '_mrkr.i
  kc=subwrd(_colors,j)
  'set ccolor 'kc
  var=subwrd(_vars,i)
  rc=getemp(var,_cint.i,_cmax.i,_cmin.i)
  'd 'var'.2(t+'to',stid='_stid._station')'
  res=result
* variable labels
  'set string 1 l 6'
  x=_parxmn
  y=_parymn-0.8
  if(subwrd(res,1)="No")
   'draw string 'x' 'y' No station data'
  else
   'draw string 'x' 'y' Raob data'
   'set string 'kc' l 6'
   'draw string 'x+0.8*k+1.2' 'y' 'var
  endif
 endif
 i=i+1
endwhile
* print single values
xval = 5.5
yval = _parymx - 1.0
if (subwrd(res,1)="Result")
 'draw string 'xval' 'yval' lon = 'subwrd(_lons,_x1)
 'draw string 'xval' 'yval-0.3' lat = 'subwrd(_lats,_y1)
 'draw string 'xval' 'yval-0.6' lev = 'subwrd(_levs,_z1)
 yval = 4.5-j*0.3
 'draw string 'xval' 'yval' 'subwrd(res,4)
endif
'draw ylab Height (m)'
rc = mydate()
p1 = tlabel(_t)
tl2 = p1 % ' ' % subwrd(rc,2) % ' ' % subwrd(rc,3) % ' ' % subwrd(rc,4)
tl3 = (_t-1)*_tinc
'draw title ' _snam._station '\' tl2'  tau=' tl3 ' hr'
* set yflip off by issuing vpage command and reset levels
* vpage also resets the colors
'set vpage off'
'set z '_z1' '_z2
* add tclock if selected
if (_clk!=0); rc=tclock(); endif
* print if selected
if(_mtyp>0); 'print'; endif
return

function disurf()
*  Function to display surface timelines
_disurf=0
rc = clear()
* define time and interpolation values
t1=int((_t1-1)/_nint+1.0)
t2=int((_t2-1)/_nint+2.0)
if(_t1=_t2)
 'set t 1 '_tnum
else
 'set t 't1' 't2
endif
'set z '_z1' '_z2
if(_sfp=0)
 'set string 1 c 6'
 'set strsiz 0.18'
 'draw string '_xpur/2' '_ypur/2-1' Select Surface Stations first'
 return
endif
* make plot areas for each selected surface station
left=_dimbxur+0.5
right=_timbxll-0.6
```

```
wide=right-left
ymax=_ypur-_vbh*2-0.25
ymin=1.0
'set line 0'
'draw recf 'left' 'ymin' 'right' 'ymax
* prescan variables
i=1; j=1
cn=0; s=0
while (i<=_vnum)
 gxtp=_vgxt.i
 if(gxtp=1); cn=cn+1; endif
 if(gxtp=2); s=j; endif
 if(_vtyp.i=1); j=j+1; endif
 i=i+1
endwhile
* plot variables
m=1
n=0
while(m<=_sfcnum)
 if(_picked.m=1)
* set up to plot shaded variable first
  if(_z1<_z2&_shad!=0)
    didshad=0
    doit=0
  else
    didshad=1
    doit=1
  endif
  j=1; k=1; l=0
  base=0.0; delta=0.0
  if(n=_sfp-1); base=0.5; delta=0.08; endif
  top=ymax-n*(ymax-ymin-0.5)/_sfp-(_sfp-1-n)/_sfp*1.03
  botm=top-(ymax-ymin-0.5)/_sfp-base
  high=(ymax-ymin-0.5)/_sfp+base
  'set vpage 'left' 'right' 'botm+delta' 'top+delta
  'set grads off'
  'set parea 0.7 'wide+1.7' 'base' 'high
  'set lon '_sfclon.m
  'set lat '_sfclat.m
  'set xlab off'
  'set grid off'
  if(n=_sfp-1); 'set xlab on'; endif
* plot grid values
  i=1
  while (i<=_vnum)
   if(_vtyp.i=1)
    if(i=_shad & didshad=0); doit=1; endif
    if(i!=_shad & didshad=1); doit=1; endif
    if(doit=1)
     gxtp=_vgxt.i
     if(gxtp=1&didshad=1); l=l+1; endif
     if(_z1<_z2)
      zz=subwrd(_gxtype,gxtp)
      'set gxout 'zz
     endif
     levlab=subwrd(_nam,_dlev)
     if(l>1&didshad=1)
      k=subwrd(_colors,l-1)
      'set ccolor 'k
     endif
     if(cn>1&j>1)
      k=subwrd(_colors,j-1)
      'set ccolor 'k
     endif
     'set cthick 6'
     if(_styl.i=4); styl=5; else; styl=_styl.i; endif
     'set cstyle 'styl
     'set cmark '_mrkr.i
     var=subwrd(_vars,i)
     rc=gctemp(var,_cint.i,_cmax.i,_cmin.i)
     if (_z1=_z2)
      kc=subwrd(_colors,j)
      'set ccolor 'kc
     endif
     'd 'var
     'set string 1 l 9'
     'set strsiz '0.2-_sfp*0.01
     'draw string 1.0 'high-0.25+_sfp*0.02' '_sfcnam.m
     'set string 1 r 9'
     str="Lon = "_sfclon.m"  Lat = "_sfclat.m
     'draw string 'wide-0.5' 'high-0.25+_sfp*0.02' 'str
     j=j+1
     doit=0
    endif
   endif
   if(_z1<_z2 & i=_shad & didshad=0)
    'set gxout contour'
    'set clevs 'levels
    'set ccolor 0'
    'd 'var
    didshad=1
    i=0
   endif
   i=i+1
  endwhile
* plot surface data
  i=1
  while (i<=_vnum)
   if(_vtyp.i=1)
    'set cthick 6'
```

```
    if(_styl.i=4); styl=5; else; styl=_styl.i; endif
    'set cstyle 'styl
    'set cmark '_mrkr.i
    var=subwrd(_vars,i)
    rc=getemp(var,_cint.i,_cmax.i,_cmin.i)
    kc=subwrd(_colors,j)
    if (_z1=_z2)
     'set ccolor 'kc
     'd 'var'.'_fn'(stid='_sfcnam.n')'
    endif
    res.i=subwrd(result,3)
    j=j+1
   endif
   i=i+1
  endwhile
* add labels
  if(n=_sfp-1)
   'set vpage off'
   'set string 1 c 10'
   'set strsiz 0.25'
   if(_z1=_z2)
    'draw string 5.5 '_ypur-_vbh*2-0.5' Surface Time Line'
   else
    'draw string 5.5 '_ypur-_vbh*2-0.5' Model Time Section'
   endif
   x=_parxmn
   y=0.6
   'set string 1 l 6'
   'set strsiz 0.12'
   'draw string 'x' 'y' Grid data'
   i=1; j=1
   while (i<=_vnum)
    if(_vtyp.i=1)
     var=subwrd(_vars,i)
     if (_z1<_z2)
      if(_shad=0)
       if(j=1)
        kc=1
       else
        kc=subwrd(_colors,j-1)
       endif
      else
       if(j<s)
        kc=subwrd(_colors,j)
       endif
       if(j=s)
        kc=1
       endif
       if(j>s)
        kc=subwrd(_colors,j-1)
       endif
      endif
     else
      kc=subwrd(_colors,j)
     endif
     'set string 'kc' l 6'
     'draw string 'x+0.8*j+1.2' 'y' 'var
     j=j+1
    endif
    i=i+1
   endwhile
   if(_z1=_z2)
    y=0.4
    'set string 1 l 6'
    'draw string 'x' 'y' Surface data'
    i=1; k=1
    while (i<=_vnum)
     if(_vtyp.i=1)
      var=subwrd(_vars,i)
      kc=subwrd(_colors,j)
      'set string 'kc' l 6'
      if(res.i="Error:")
       'set string 1 l 6'
       'draw string 'x+0.8*k+1.2' 'y' none'
      else
       'draw string 'x+0.8*k+1.2' 'y' 'var
      endif
      j=j+1
      k=k+1
     endif
     i=i+1
    endwhile
   endif
  endif
  n=n+1
 endif
 m=m+1
endwhile
* vpage also resets the colors
'set vpage off'
'set x '_x1' '_x2
'set y '_y1' '_y2
_t = _t1
if(_mtyp>0); 'print'; endif
return

function vecvar(var,gxtp,n,a,b,t1,t2)
* Define vector variable
nchvar=_nchvr
varz=substr(var,2,nchvar)
```

```
uu='u'%varz
vv='v'%varz
* define vecvar
if(b=0)
 if(gxtp=3|gxtp=4)
  vecvar='skip('uu','n');'vv';mag('uu','vv')'
 else
  vecvar='mag('uu','vv')'
 endif
 if(gxtp=5)
  vecvar=uu';'vv';mag('uu','vv')'
 endif
else
 arg1=a'*'uu'+'b'*'uu'(t+1)'
 arg2=a'*'vv'+'b'*'vv'(t+1)'
 if(gxtp=3|gxtp=4)
  vecvar='skip('arg1','n');'arg2';mag('arg1','arg2')'
 else
  vecvar='mag('arg1','arg2')'
 endif
 if(gxtp=5)
  vecvar=arg1';'arg2';mag('arg1','arg2')'
 endif
endif
*set display attributes
rc=getemp(u,_cint._uvar,_cmax._uvar,0)
'set arrscl 0.5 '_cmax._uvar
'set strmden '_dens
return vecvar

function howlong(str)
* Find string length
i=1
while (z != "")
 z=substr(str,i,1)
 i=i+1
endwhile
return i

function plottype()
* Set plot type
* plottyp=0 is no plot, print data
* plottyp=1 is line plot, static and animated
* plottyp=2 is 2D section, static and animated
* plottyp=3 is line plot for displayed dimension range
* plottyp=4 is static time section
* plottyp=5 is animated time section
* plottyp=6 is line plot for displayed time range
* plottyp=7 is 4D, no plot
if(_t1=_t2&_x1=_x2&_y1=_y2&_z1=_z2)
 _plottyp=0
endif
if(_t1=_t2&_x1=_x2&_y1=_y2&_z1<_z2)
 if (_styp=3); _plottyp=3; else; _plottyp=1; endif
endif
if(_t1=_t2&_x1=_x2&_y1<_y2&_z1=_z2)
 if (_styp=2); _plottyp=3; else; _plottyp=1; endif
endif
if(_t1=_t2&_x1=_x2&_y1<_y2&_z1<_z2)
 if (_styp=1); _plottyp=2; else; _plottyp=1; endif
endif
if(_t1=_t2&_x1<_x2&_y1=_y2&_z1=_z2)
 if (_styp=1); _plottyp=3; else; _plottyp=1; endif
endif
if(_t1=_t2&_x1<_x2&_y1=_y2&_z1<_z2)
 if (_styp=2); _plottyp=2; else; _plottyp=1; endif
endif
if(_t1=_t2&_x1<_x2&_y1<_y2&_z1=_z2)
 if (_styp=3); _plottyp=2; else; _plottyp=1; endif
endif
if(_t1=_t2&_x1<_x2&_y1<_y2&_z1<_z2)
 _plottyp=2
endif
if(_t1<_t2&_x1=_x2&_y1=_y2&_z1=_z2)
 _plottyp=6
endif
if(_t1<_t2&_x1=_x2&_y1=_y2&_z1<_z2)
 if (_styp=3); _plottyp=4; else; _plottyp=1; endif
endif
if(_t1<_t2&_x1=_x2&_y1<_y2&_z1=_z2)
 if (_styp=2); _plottyp=4; else; _plottyp=1; endif
endif
if(_t1<_t2&_x1=_x2&_y1<_y2&_z1<_z2)
 if (_styp=1); _plottyp=2; else; _plottyp=5; endif
endif
if(_t1<_t2&_x1<_x2&_y1=_y2&_z1=_z2)
 if (_styp=1); _plottyp=4; else; _plottyp=1; endif
endif
if(_t1<_t2&_x1<_x2&_y1=_y2&_z1<_z2)
 if (_styp=2); _plottyp=2; else; _plottyp=5; endif
endif
if(_t1<_t2&_x1<_x2&_y1<_y2&_z1=_z2)
 if (_styp=3); _plottyp=2; else; _plottyp=5; endif
endif
if(_t1<_t2&_x1<_x2&_y1<_y2&_z1<_z2)
 _plottyp=7
endif
return

function style(i)
```

```
* Set line plot style
 if(_styl.i=1); _sty.i='solid'; endif
 if(_styl.i=2); _sty.i='long dash'; endif
 if(_styl.i=3); _sty.i='short dash'; endif
 if(_styl.i=4); _sty.i='dotted'; endif
return

function marker(i)
* Set data marker style
 if(_mrkr.i=0); _mkr.i='none'; endif
 if(_mrkr.i=1); _mkr.i='cross'; endif
 if(_mrkr.i=2); _mkr.i='open circle'; endif
 if(_mrkr.i=3); _mkr.i='closed circle'; endif
 if(_mrkr.i=4); _mkr.i='open square'; endif
 if(_mrkr.i=5); _mkr.i='closed square'; endif
return

function getemp(vname,cint,cmax,cmin)
*  Set contour limits and increments
** WARNING!!! (ce-co)/ci must be < 99 **
 ci=10; co=-100; ce=100; cinc=5
 if(vname="dmdz"); ci=40; co=-200; ce=200; cinc=5; endif
 if(vname="ee"); ci=10; co=0; ce=100; cinc=10; endif
 if(vname="eh"); ci=10; co=0; ce=100; cinc=10; endif
 if(vname="hdiv"); ci=0.00001; co=-0.0004; ce=0.0004; cinc=0.00001; endif
 if(vname="m"); ci=25; co=250; ce=1000; cinc=5; endif
 if(vname="och"); ci=50; co=0; ce=3000; cinc=5; endif
 if(vname="p"); ci=50; co=0; ce=1050; cinc=5; endif
 if(vname="pc"); ci=0.1; co=0; ce=10; cinc=0.02; endif
 if(vname="ps"); ci=5000; co=60000; ce=105000; cinc=500; endif
 if(vname="pmsl"); ci=2; co=960; ce=1030; cinc=1; endif
 if(vname="q"); ci=1; co=0; ce=30; cinc=0.5; endif
 if(vname="qc"); ci=0.000001; co=0; ce=0.00005; cinc=0.0000005; endif
 if(vname="qi"); ci=0.000001; co=0; ce=0.00005; cinc=0.0000005; endif
 if(vname="qr"); ci=0.000001; co=0; ce=0.00005; cinc=0.0000005; endif
 if(vname="rh"); ci=5; co=0; ce=100; cinc=1; endif
 if(vname="ri"); ci=0.5; co=-5; ce=1; cinc=0.1; endif
 if(vname="slp"); ci=400; co=96000; ce=103000; cinc=100; endif
 if(vname="tau"); ci=0.02; co=0; ce=1; cinc=0.01; endif
 if(vname="t"); ci=2; co=250; ce=310; cinc=1; endif
 if(vname="tc"); ci=2; co=-50; ce=40; cinc=1; endif
 if(vname="td"); ci=2; co=-50; ce=40; cinc=1; endif
 if(vname="terr"); ci=200; co=0; ce=5000; cinc=10; endif
 if(vname="th"); ci=1; co=280; ce=330; cinc=1; endif
 if(vname="ts"); ci=1; co=280; ce=330; cinc=1; endif
 if(vname="u"); ci=1; co=-30; ce=30; cinc=1; endif
 if(vname="v"); ci=1; co=-30; ce=30; cinc=1; endif
 if(vname="vort"); ci=0.00001; co=-0.0004; ce=0.0004; cinc=0.00001; endif
 if(vname="vv"); ci=1; co=0; ce=10; cinc=0.5; endif
 if(vname="z"); ci=500; co=-5000; ce=5000; cinc=10; endif
 if(vname="z5"); ci=500; co=4500; ce=6500; cinc=10; endif
 if(cint!="")
  ci=cint
 endif
 _ci=ci
 _cinc=cinc
 if(cmax!="")
  ce=cmax
 endif
 _cmx=ce
 if(cmin!="")
  co=cmin
 endif
 _cmn=co
 i = 0
 levels=co
 cc = co
 while (cc < ce)
  i = i + 1
  cc=co + i*ci
  levels=levels % ' ' % cc
 endwhile
 'set clevs 'levels
 'set vrange 'co' 'ce
 if (i>=99)
  say 'Variable 'vname' has 'i+1' levels:'
  say levels
 endif
return levels

function readsfc()
* Read surface station data
 i=0
 while (1)
  res=read(stations.sfc)
  dum=sublin(res,1)
  cod=subwrd(dum,1)
  if(cod!=0)
   if(cod=1); say "Error opening file"; endif
   if(cod=8); say "File open for write"; endif
   if(cod=9); say "I/O error"; endif
   break
  endif
  i=i+1
  dum=sublin(res,2)
  _sfcnam.i=subwrd(dum,1)
  _sfclat.i=subwrd(dum,2)
  _sfclon.i=subwrd(dum,3)
 endwhile
 _sfcnum=i
```

```
 rc=close (stations.sfc)
return

function readua()
* Read upper air (raob) station data
 i=0
 while (1)
  res=read(stations.ua)
  dum=sublin(res,1)
  cod=subwrd(dum,1)
  if(cod!=0)
   if(cod=1); say "Error opening file"; endif
   if(cod=8); say "File open for write"; endif
   if(cod=9); say "I/O error"; endif
   break
  endif
  i=i+1
  dum=sublin(res,2)
  _stid.i=subwrd(dum,1)
  _slat.i=subwrd(dum,2)
  _slon.i=subwrd(dum,3)
  _snam.i=subwrd(dum,4)
  _snam.i=_snam.i %' '% subwrd(dum,5)
  _snam.i=_snam.i %' '% subwrd(dum,6)
 endwhile
 _stnnum=i
 rc=close (stations.ua)
return

function getloc(lat,lon)
* Determine the location of selection surface station
 test=1
 pick=0
 i=0
 while(i<_sfcnum)
  i=i+1
  lat2=(_sfclat.i-lat)*(_sfclat.i-lat)
  lon2=(_sfclon.i-lon)*(_sfclon.i-lon)
  sum2=lat2+lon2
  if(sum2<test)
   test=sum2
   pick=i
  endif
 endwhile
return pick

function picksfc(x,y)
* Determine selected surface station
 'q xy2w 'x' 'y
 lon=subwrd(result,3)
 lat=subwrd(result,6)
 n=getloc(lat,lon)
 if(n=0); return; endif
 if(_picked.n!=1); i=1; endif
 if(_picked.n=1); i=0; endif
 if(i=1)
  'q w2xy '_sfclon.n' '_sfclat.n
  _sfx.n=subwrd(result,3)
  _sfy.n=subwrd(result,6)
  'set line 31'
  'draw mark 3 '_sfx.n' '_sfy.n' 0.08'
  'set string 31 bl 1'
  'set strsiz 0.08'
  'draw string '_sfx.n+0.08' '_sfy.n+0.08' '_sfcnam.n
  _picked.n=1
  _sfp=_sfp+1
 endif
 if(i=0 & _sfp!=0)
  'set line 0'
  'draw mark 3 '_sfx.n' '_sfy.n' 0.08'
  'set line 30'
  'draw mark 2 '_sfx.n' '_sfy.n' 0.08'
  'set string 0 bl 1'
  'set strsiz 0.08'
  'draw string '_sfx.n+0.08' '_sfy.n+0.08' '_sfcnam.n
  _picked.n=0
  _sfp=_sfp-1
 endif
return

function output()
* Control print output
 rc = clear()
 'set strsiz 0.18'
 'set string 2 c 12'
 if(_nfram>1&_mtyp=2)
  'set strsiz 0.225'
  str='ONLY ONE COLOR FRAME CAN BE PRINTED'
  'draw string '_xpur/2' '_ypur/2' 'str
  'set strsiz 0.18'
  str='Eliminate any animations and reprint'
  'draw string '_xpur/2' '_ypur/2-1' 'str
  str='Click any mouse button to continue'
  'draw string '_xpur/2' '_ypur/2-3' 'str
  'q pos'
 else
  if(_mtyp=1); str='BLACK AND WHITE PLOTS'; endif
  if(_mtyp=2); str='COLOR PLOTS'; endif
  if(_mtyp=3); str='GrADS PLOT IMAGES'; endif
  if(_mtyp=4); str='VIDEO IMAGES'; endif
```

```
 'draw string '_xpur/2' '_ypur/2' '_nfram' 'str' WILL BE PRINTED'
 'set button 2 1 91 92 6'
 'draw button 98 '_xpur/2-1' '_ypur/2-1' 1.5 0.8 CONTINUE'
 'draw button 99 '_xpur/2+1' '_ypur/2-1' 1.5 0.8 CANCEL'
 'q pos'
 btn = subwrd(result,7)
 if (btn=98)
  rc = clear()
  file='otype.'_name
  rc=write (file, _mtyp)
  rc=write (file, _type, append)
  rc=close (file)
  'run output.gs'
  '!rm 'file
 endif
 rc = clear()
endif
'!rm browse.*.'_name
return

function zoomin()
* Perform zoom-in calculations
 'q xy2gr '_xpos' '_ypos
 xcen=subwrd(result,3)
 ycen=subwrd(result,6)
 xrng=_x2-_x1
 yrng=_y2-_y1
 xmid=(_x2+_x1)/2
 ymid=(_y2+_y1)/2
 xfrac=(xcen-xmid)/xrng
 yfrac=(ycen-ymid)/yrng
 if(xfrac<0); xfrac=-xfrac; endif
 if(yfrac<0); yfrac=-yfrac; endif
 if ((xfrac<0.1&yfrac<0.1)|(_x1=1&_x2=_xnum&_y1=1&_y2=_ynum))
  _x2=int(xcen+xrng/4+0.51)
  _y2=int(ycen+yrng/4+0.51)
  _x1=int(xcen-xrng/4+0.49)
  _y1=int(ycen-yrng/4+0.49)
 else
  _x2=int(xcen+xrng/2+0.51)
  _y2=int(ycen+yrng/2+0.51)
  _x1=int(xcen-xrng/2+0.49)
  _y1=int(ycen-yrng/2+0.49)
 endif
 if(_x2>_xnum); _x2=_xnum; endif
 if(_y2>_ynum); _y2=_ynum; endif
 if(_x1<1); _x1=1; endif
 if(_y1<1); _y1=1; endif
return

function zoomout()
* Perform zoom-out calculations
 xrng=_x2-_x1
 yrng=_y2-_y1
 xmid=(_x2+_x1)/2
 ymid=(_y2+_y1)/2
 _x2=int(xmid+xrng+0.51)
 _y2=int(ymid+yrng+0.51)
 _x1=int(xmid-xrng+0.49)
 _y1=int(ymid-yrng+0.49)
 if(_x2>_xnum); _x2=_xnum; endif
 if(_y2>_ynum); _y2=_ynum; endif
 if(_x1<1); _x1=1; endif
 if(_y1<1); _y1=1; endif
return

function clear()
* Clear screen
 'clear'
 'set line 95'
 'draw recf 0.0 0.0 '_xpur' '_ypur
 _display=0
return

function colors()
* Set and compensate display colors
* get $DISPLAY
 '!echo $DISPLAY > display'
 rc = read (display)
 type = sublin(rc,2)
 rc = close(display)
 i = 1
 while (char!=":")
  char = substr(type,i,1)
  i = i + 1
 endwhile
 name = substr(type,1,i-1)
 if(name=":")
  '!echo $USER > display'
  rc = read (display)
  _name = sublin(rc,2)
  rc = close(display)
 else
  _name = substr(name,1,i-2)
 endif
 '!rm display'
 '!grep '"'_name'"' /etc/hosts > display'
 i=0; j=0
 while (j=0)
  i=i+1
```

```
 rc = read(display)
 info = sublin(rc,2)
 temp = subwrd(info,2)
 tmp1 = substr(temp,1,2)
 if(tmp1='pc'); type=tmp1; endif
 if(temp=_name)
  temp = subwrd(info,3)
  tmp2 = substr(temp,1,2)
  if(tmp2='gr'|tmp2='hx'|tmp2='sn'); type=tmp2; j=1; endif
 endif
 if(i=4); j=1; endif
endwhile
rc = close(display)
'!rm display'
* set factor
fac=1.0; add=0.0
if(type='gr'); fac=1.0; add=0; endif
if(type='hx'); fac=1.1; add=15; endif
if(type='pc'); fac=1.1; add=63; endif
if(type='sn'); fac=1.1; add=63; endif
_type=type
say 'DISPLAY='name' type='type
* set colors
cn = (255-add)*100*fac/255+add
tl = (255-add)*50*fac/255+add
br = (255-add)*200*fac/255+add
bg = (255-add)*25*fac/255+add
* set button colors: 90-center; 91-top,left; 92-bottom,right
'set rgb 90 'cn' 'cn' 'cn
'set rgb 91 'tl' 'tl' 'tl
'set rgb 92 'br' 'br' 'br
* background color
'set rgb 95 0 0 'bg
* set plot, label, and other colors
_colors="10 8 5 6 14 3 11 12 9 7 13 15 10 8 5 6 14 3 11 12 9 7 13 15"
'set rgb 30 159 255 255'
'set rgb 31 255 53 159'
'set rgb 40 64 64 64'
'set rgb 41 255 255 0'
return

function tclock()
* Control time clock
x0 = (_timbxll+_timbxur)/2-0.15
y0 = (_timbyll+_timbyur)/2
csize = 2*(_timbxur-_timbxll)
* get greenwich time
rc = mydate()
day=subwrd(rc,2) % ' ' % subwrd(rc,3)
rc = tlabel(_t)
 time=subwrd(rc,1)
 hour=substr(time,1,2)
 if (substr(time,3,1)=":");
  min=substr(time,4,2)
  if(substr(time,6,1)=":"); min=min+substr(time,7,2)/60; endif
 else
  min=0
 endif
* correct to local
 xmid=int((_x2+_x1)/2)
 lmid=subwrd(_lons,xmid)
 if(lmid<0)
  gmoffset=int(lmid/15-0.5)
 else
  gmoffset=int(lmid/15+0.5)
 endif
 hour=hour+gmoffset
 if(hour<0); hour= hour+24; endif
 if(hour>=24); hour=hour-24; endif
 if(hour>=6&hour<=19)
  cfacecol=7
 else
  cfacecol=0
 endif
 if(hour<12)
  ampm='AM'
 else
  ampm='PM'
 endif
 rc=clock(x0,y0,csize,hour,min,cfacecol,ampm,day)
return

function clock(x0,y0,csize,hour,min,cfacecol,ampm,day)
* Function to draw a clock with hour and min hands
 pi2=2*3.14159
 radius=csize*0.5
 hourhand=radius*0.6
 minhand=radius*0.8
 handcol=1
 if(cfacecol!=0);handcol=0;endif
 'set line 'cfacecol
 'draw mark 3 'x0' 'y0' 'csize
 'set line 1 1 8'
 'draw mark 2 'x0' 'y0' 'csize
* draw the min and hour hands
 thetamin=(min/60)*pi2
 thetahour=((hour+min/60)/12)*pi2
 'd sin('thetamin')'
 xm=subwrd(result,4)
 xm=xm*minhand
```

```
'd cos('thetamin')'
ym=subwrd(result,4)
ym=ym*minhand
'd sin('thetahour')'
xh=subwrd(result,4)
xh=xh*hourhand
'd cos('thetahour')'
yh=subwrd(result,4)
yh=yh*hourhand
x1=x0+xm
y1=y0+ym
'set line 'handcol' 1 5'
'draw line 'x0' 'y0' 'x1' 'y1
x1=x0+xh
y1=y0+yh
'set line 'handcol' 1 6'
'draw line 'x0' 'y0' 'x1' 'y1
ys=y0-radius-0.1
'set strsiz 0.10'
'set string 3 c 4'
'draw string 'x0' 'ys' 'ampm' 'day
'draw string 'x0' 'ys-0.175' LOCAL'
return
```

B.2. OUTPUT.GS

```
function output(args)
*
*  Script to direct output of NRL Data Browser
*
*  This script allows the customization of the browser output
*  without modifying the browser script.  The output path, device
*  or software can be changed in the shell command lines that start
*  with '!
*
*  Define page limits
*
 'q gxinfo'
 dum=sublin(result,2)
 _xpur=subwrd(dum,4)
 _ypur=subwrd(dum,6)
*
*  Determine output
 name=subwrd(args,1)
 term=subwrd(args,2)
 what=subwrd(args,3)
 if(term='sn'); prt=lpr; else; prt=lp; endif
say 'local is 'name' using 'prt' on 'term
*
* Set string, button, and color characteristics
 'set strsiz 0.2'
 'set string 1 c 12'
 'set button 0 90 92 91 7'
 'set button 1 90 91 92 7'
 'set rgb 90 127 127 127'
 'set rgb 91 63 63 63'
 'set rgb 92 223 223 223'
*
* Direct output based on where
 site1='Local'
 site2='702'
 site3='704'
 site4='File'
* Define output file type
 type1='Laser'
 type2='PS'
 type3='Gif'
 type4='Tiff'
* Define output media type
 media1='Hardcopy'
 media2='Transparency'
* Determine output file path and name
 '!pwd > dir'
 rc = read(dir)
 wdir=sublin(rc,2)
 rc = close(dir)
 '!rm dir'
 file='nebs.prn.'name
*
 if (what=1)
  'draw string '_xpur/2' '_ypur/2' PRINTING BLACK AND WHITE PLOTS'
  '!gxps -i browse.prn.'name' -o browse.ps.'name
  'draw button 30 '_xpur/2-3' '_ypur/2-1' 0.8 0.5 'site1
  'draw button 31 '_xpur/2-1' '_ypur/2-1' 0.8 0.5 'site2
  'draw button 32 '_xpur/2+1' '_ypur/2-1' 0.8 0.5 'site3
  'draw button 33 '_xpur/2+3' '_ypur/2-1' 0.8 0.5 'site4
  'q pos'
  btn = subwrd(result,7)
* PUT YOUR B&W PRINTER PATH AND COMMAND HERE
  if (btn=30)
   if(name='siquig')
    '!cat browse.ps.'name' | rsh 'name' lpr -h'
   else
    '!pstolj browse.ps.'name' | rsh 'name' 'prt
   endif
  endif
  if (btn=31)
   '!cat browse.ps.'name' | rsh krypton lpr -h -P admin-ps'
  endif
  if (btn=32)
   '!cat browse.ps.'name' | rsh krypton lpr -h -P post'
  endif
  if (btn=33)
   'draw button 30 '_xpur/2-3' '_ypur/2-1' 0.8 0.5 'type1
   'draw button 31 '_xpur/2-1' '_ypur/2-1' 0.8 0.5 'type2
   'draw button 32 '_xpur/2+1' '_ypur/2-1' 0.8 0.5 'type3
   'draw button 33 '_xpur/2+3' '_ypur/2-1' 0.8 0.5 'type4
   'q pos'
   btn = subwrd(result,7)
   where=wdir'/'file
   'set strsiz 0.15'
   if (btn=30)
    say 'Saving printer file 'where
    'draw string '_xpur/2' '_ypur/2-2' SAVING PRINTER FILE 'file
    '!pstolj browse.ps.'name' > 'file
   endif
   if (btn=31)
    say 'Saving Post Script file 'where'.ps'
    'draw string '_xpur/2' '_ypur/2-2' SAVING POSTSCRIPT FILE 'file'.ps'
    '!mv browse.ps.'name' 'file'.ps'
   endif
   if (btn=32)
    say 'Saving Gif file 'where'.gif'
    'draw string '_xpur/2' '_ypur/2-2' SAVING GIF FILE 'file'.gif'
```

```
   '!pstogif browse.ps.'name' 'file'.gif 150'
   endif
   if (btn=33)
    say 'Saving Post Script file 'where'.tiff'
    'draw string '_xpur/2' '_ypur/2-2' SAVING TIFF FILE 'file'.tiff'
    '!pstotiff browse.ps.'name' 'file'.tiff 150'
   endif
  endif
  say "Black and White plots printed."
 endif
 if (what=2)
  'draw string '_xpur/2' '_ypur/2' PRINTING COLOR PLOTS'
* temporary patch for broken gxpscw
*   '!gxpsc -i browse.prn.'name' -o browse.ps.'name
*   '!pstogif browse.ps.'name' 'file'.gif 150'
*   '!fromgif 'file'.gif 'file'.rgb'
*   '!invbw 'file'.rgb 'file'.rbw'
*   '!tops 'file'.rbw -rgb > browse.ps.'name
*   '!rm 'file'.gif 'file'.rgb 'file'.rbw'
  '!gxpscw -i browse.prn.'name' -o browse.ps.'name
  'draw button 30 '_xpur/2-3' '_ypur/2-1' 0.8 0.5 'site1
  'draw button 31 '_xpur/2-1' '_ypur/2-1' 0.8 0.5 'site2
  'draw button 32 '_xpur/2+1' '_ypur/2-1' 0.8 0.5 'site3
  'draw button 33 '_xpur/2+3' '_ypur/2-1' 0.8 0.5 'site4
  'q pos'
  btn = subwrd(result,7)
* PUT YOUR COLOR PRINTER PATH AND COMMAND HERE
  if (btn=30)
  endif
  if (btn=31)
   'clear'
   'set line 95'
   'draw recf 0.0 0.0 '_xpur' '_ypur
   'draw button 34 '_xpur/2-1' '_ypur/2-1' 1.5 0.5 'media1
   'draw button 35 '_xpur/2+1' '_ypur/2-1' 1.5 0.5 'media2
   'q pos'
   btn = subwrd(result,7)
   if (btn=34)
   '!cat browse.ps.'name' | rsh krypton lpr -h -P color'
   endif
   if (btn=35)
   '!cat browse.ps.'name' | rsh krypton lpr -h -P color_trans'
   endif
  endif
  if (btn=32)
   '!pstorast browse.ps.'name' | rsh krypton lpr -v -h -P tcprSRF'
  endif
  if (btn=33)
*    '!gxpsc -i browse.prn.'name' -o browse.ps.'name
   'draw button 30 '_xpur/2-3' '_ypur/2-1' 0.8 0.5 'type1
   'draw button 31 '_xpur/2-1' '_ypur/2-1' 0.8 0.5 'type2
   'draw button 32 '_xpur/2+1' '_ypur/2-1' 0.8 0.5 'type3
   'draw button 33 '_xpur/2+3' '_ypur/2-1' 0.8 0.5 'type4
   'q pos'
   btn = subwrd(result,7)
   whcre=wdir'/'file
   'set strsiz 0.15'
   if (btn=30)
    say 'Saving printer file 'where
    'draw string '_xpur/2' '_ypur/2-2' SAVING PRINTER FILE 'file
    '!pstorast browse.ps.'name' > 'file'
   endif
   if (btn=31)
    say 'Saving Post Script file 'where'.ps'
    'draw string '_xpur/2' '_ypur/2-2' SAVING POSTSCRIPT FILE 'file'.ps'
    '!mv browse.ps.'name' 'file'.ps'
   endif
   if (btn=32)
    say 'Saving Gif file 'where'.gif'
    'draw string '_xpur/2' '_ypur/2-2' SAVING GIF FILE 'file'.gif'
    '!pstogif browse.ps.'name' 'file'.gif 150'
   endif
   if (btn=33)
    say 'Saving Post Script file 'where'.tiff'
    'draw string '_xpur/2' '_ypur/2-2' SAVING TIFF FILE 'file'.tiff'
    '!pstotiff browse.ps.'name' 'file'.tiff 150'
   endif
  endif
  say "Color plots printed."
 endif
 if (what=3)
   'draw string '_xpur/2' '_ypur/2' MAKING GrADS PLOT IMAGES'
   '!run_gxtran browse.prn.'name
   say "GrADS plot images sent to viewer."
 endif
 if (what=4)
   'draw string '_xpur/2' '_ypur/2' MAKING VIDEO IMAGES'
   '!gxpsc -i browse.prn.'name' -o browse.ps.'name
   say "Plot images sent to video recorder."
 endif
return
```